VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTOMATIZOVANÝ SYSTÉM PRO SKENOVÁNÍ KONSTRUKČNÍCH DÍLŮ
AUTOMATED SYSTEM FOR COMPONENTS SCANNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jan Hřib**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Miroslav Jirgl, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Jan Hřib ***ID:*** 186089
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Automatizovaný systém pro skenování konstrukčních dílů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a realizovat automatický systém pro skenování konstrukčních dílů pro účely inspekce jejich rozměrů a tolerancí. Předpokladem je výběr vhodného HW a SW prostředků pro zpracování 3D dat.

1. Seznamte se s technologií a stručně popište.
2. Navrhněte HW koncepci a vyberte vhodné komponenty.
3. Seznamte se s knihovnou Point Cloud Library (PCL) a stručně popište její vlastnosti a možnosti použití.
4. Navrhněte a implementujte SW řešení.
5. Otestujte a demonstrujte funkčnost.

DOPORUČENÁ LITERATURA:

Point Cloud Library (PCL) [online]. [cit. 2020-09-14]. Dostupné z: https://pointclouds.org/

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

Abstrakt

Cílem této diplomové práce je návrh automatizovaného systému pro skenování konstrukčních dílů pro účely inspekce jejich rozměrů a tolerancí. Teoretický úvod poskytuje čtenáři základní informace týkající se tématu 3D skenování. Práce obsahuje rovněž návrhu vlastní skenovací sestavy. Největší pozornost je věnována návrhu programu využívajícím knihovnu PCL. Cílem programu je automatické zpracování dat ze 3D skeneru a vyhodnocení požadovaných rozměrů naskenovaného konstrukčního dílu. Závěrečná část práce se věnuje testování navrženého řešení.

Klíčová slova

Point Cloud Library, PCL, C++, 3D skener, inspekce rozměrů, automatické měření

Abstract

The aim of this thesis is to design an automated scanning system for components for the purpose of inspecting their dimensions and tolerances. The theoretical introduction provides the reader with basic information on the topic of 3D scanning. The work also includes the design of own scanning system. The greatest attention is paid to the design of a program using the PCL library. The aim of the program is automatic processing of data from a 3D scanner and evaluation of the required dimensions of the scanned component. The final part of the work is devoted to testing the proposed solution.

Keywords

Point Cloud Library, PCL, C++, 3D scanner, inspection of dimensions, automatic evaluation

Bibliografická citace

HŘIB, Jan. *Automatizovaný systém pro skenování konstrukčních dílů.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/130785. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miroslav Jirgl.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Jan Hřib*
VUT ID studenta:	*186089*
Typ práce:	*Diplomová práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	Automatizovaný systém pro skenování konstrukčních dílů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 14. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Miroslavu Jirglovi, Ph.D. za účinnou metodickou a pedagogickou pomoc a další cenné rady při zpracování mé diplomové práce. Dále děkuji firmě Acam Solution za poskytnutí zázemí pro realizaci této práce, jmenovitě pak Ing. Lukáši Honcovi za cenné odborné rady.

V Brně dne: 14. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Průmyslová automatizace hraje v dnešním světě důležitou roli. Dotýká se téměř všech oblastí od energetiky, potravinářství až po výrobu nejrůznějších dílů nebo kompletování složitějších sestav. Díky automatizaci procesu je totiž možné docílit lepších výsledků za nižší provozní náklady. Součástí automatizovaného procesu bývá na jeho konci i fáze, kdy vyráběný produkt prochází kontrolou. Skutečné parametry výstupního produktu jsou za účelem dodržení kvality výroby porovnávány s jeho požadovanými vlastnostmi. Produktem můžeme rozumět cokoliv od surovin zpracovávaných v potravinářském průmyslu, přes produkty petrochemického průmyslu až po nejrůznější konstrukční díly. Tato práce se omezuje právě na poslední zmíněné a bude se věnovat navržení a realizaci automatického systému pro skenování konstrukčních dílů pro účely inspekce jejich rozměrů a tolerancí.

Pořizování dat obsahujících informaci o rozměrech snímaného prostoru nebo objektu je v dnešní době sám o sobě široce využívaná technologie. 3D skenování nachází své využití v lékařství, stavitelství, architektuře, výrobním průmyslu, logistice, reverzním inženýrství a v mnoha dalších odvětvích. Existuje více způsobů, jak lze 3D data získávat, čemuž je věnována úvodní kapitola této práce.

Po přehledu možností získávání 3D dat se práce zaměřuje na návrh skenovací sestavy, která bude sloužit k realizaci testovacího měření. Komponenty pro realizaci sestavy poskytla firma Acam Solution, která je zároveň zadavatelem tématu této práce.

Následující kapitoly se věnují softwarové stránce problematiky. Na trhu sice existují softwarové nástroje, které slouží k inspekci rozměrů naskenovaných předmětů, ale tyto programy nejsou často připraveny na zapojení do libovolného automatického systému a většinou také pro svou funkčnost vyžadují lidskou obsluhu. Pro realizaci vlastního softwarového nástroje byla vybrána C++ knihovna Point Cloud Library.

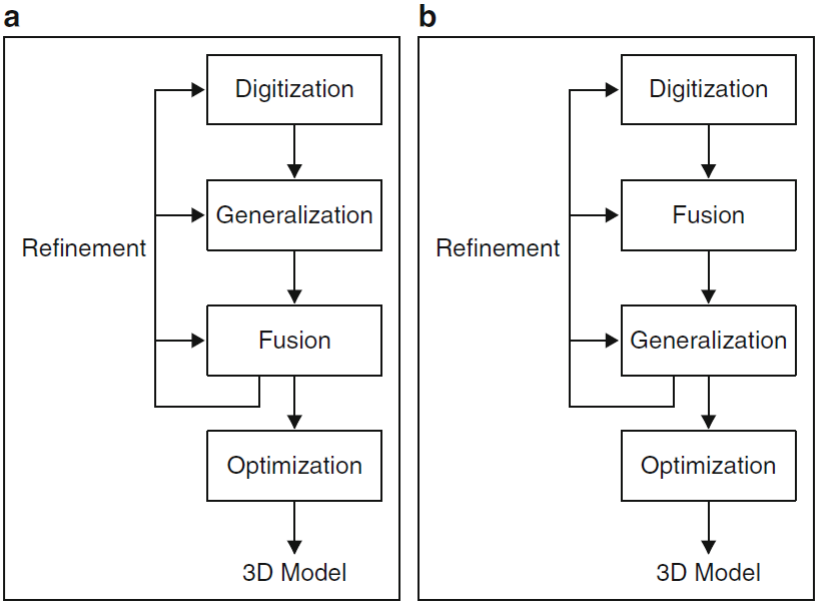
Na závěr práce byla provedena série testovacích měření, která mají za úkol ověřit funkčnost navrženého řešení. Výsledek práce poslouží firmě Acam Solution při plánovaném vývoji vlastního automatizovaného systému měření rozměrů konstrukčních dílů využívajícím technologii 3D skenování.

1. PŘEHLED TECHNOLOGIE

Úvodní kapitola práce se zabývá obecným přehledem technologie 3D skenování. V následujících kapitolách budou popsány jednotlivé principy fungování skenerů, výhody a nevýhody těchto zařízení a také přehled způsobů reprezentace 3D dat.

1.1 Proces 3D skenování

Zařízení sloužící k získávání dat, která obsahují nejen prostorové informace o reálném objektu, se obecně nazývá 3D skener. V závislosti na funkčním principu použitého skeneru mohou data obsahovat informaci například i o barvě nebo textuře povrchu [1]. Nezávisle na typu 3D skeneru můžeme rozdělit proces získávání 3D dat na kroky uvedené na obrázku 1.1. Varianta *a* představuje postup, kdy surová data z jednotlivých skenů nejprve zpracujeme a následně z nich sloučením vytvoříme výsledný model, ve variantě *b* veškeré surové skeny nejprve sloučíme a ty následně zpracujeme požadovaným způsobem. Krok slučování více skenů do jednoho výsledného souboru však není nutné provádět ve všech případech. Například pro účely této práce se nepočítá s využitím sjednocování dílčích skenů do jednoho souboru.

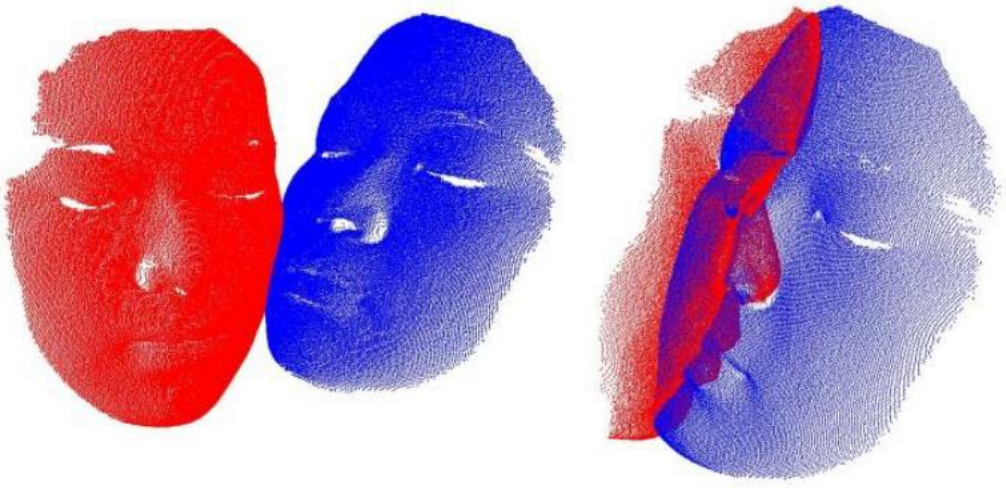


Obrázek 1.1 Obecné schéma procesu 3D skenování [1].

Během kroku digitalizace skener pořídí surová data, která jsou interpretována jako mračna bodů, neboli anglicky point cloudy. Pořízená data představují digitální prostorovou reprezentaci daného objektu, která může být v některých případech doplněna o další informace, jako je například barva povrchu. V následujícím kroku – Generalization – dochází v případě potřeby k rekonstrukci povrchu naskenovaného předmětu. Obyčejné proložení bodů nedosahuje dobrých výsledků, ale pro předběžný

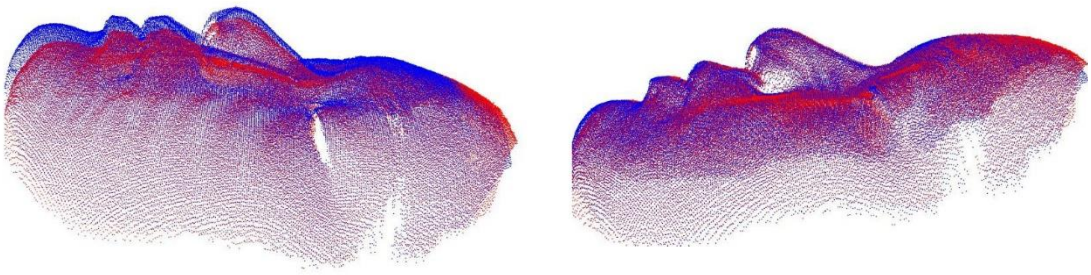
odhad, který následně zpřesníme, lze použít. Surová naskenovaná data zpravidla obsahují šum, případně neplatné body vzniklé chybami hardwaru nebo vlivem prostředí. Těchto nežádoucích dat se lze zbavit filtrací vstupních dat [1] [2].

Velmi často nastává situace, kdy má objekt zájmu složitější geometrii, a proto je nutné provést postupné skenování z více úhlů nebo lze použít více skenerů naráz. Jednotlivé skeny je nutné následně zarovnat vůči sobě, čemuž se obecně říká registrace [1]. Pro snížení výpočetní náročnosti registrace může být vytvořen alternativní model s redukovaným počtem bodů, který však stále dostatečně přesně popisuje tvar reálného objektu. Aby však při práci s daty nedocházelo ke ztrátě informací, tak se redukovaný model využívá pouze ke zjištění transformační matice, která je následně aplikována na původní detailní model s neredukovaným počtem bodů [3]. K zarovnání dvou modelů vůči sobě slouží například metoda ICP (Iterative Closest Point), při které je iterativně zmenšována odchylka mezi jednotlivými skeny až do momentu, kdy je odchylka menší než nastavený práh [4]. Hsieh ve své práci [3] však poukazuje na problém, kdy nemusí k zarovnání dojít například ani po 1000 iteracích metody ICP (viz obrázek 1.2).



Obrázek 1.2 Neúspěšný výsledek metody ICP [3]

Řešením je proces registrace rozdělit do více fází – na fázi hrubého a jemného zarovnání. Během hrubého zarovnání jsou dílčí point cloudy zarovnány jen přibližně například pomocí nalezení korespondenčních párů z jednotlivých point cloudů. Na takto připravená data je následně aplikována metoda ICP, která díky předešlému hrubému zarovnání konverguje s větší pravděpodobností ke správným výsledkům [3]. Úspěšný výsledek registrace znázorňuje obrázek 1.3.



Obrázek 1.3 Nalevo dva modely před registrací, napravo úspěšný výsledek registrace pomocí metody ICP. [3]

Po provedení registrace můžou být body z jednotlivých skenů sloučeny do jednoho výsledného point cloudu [1].

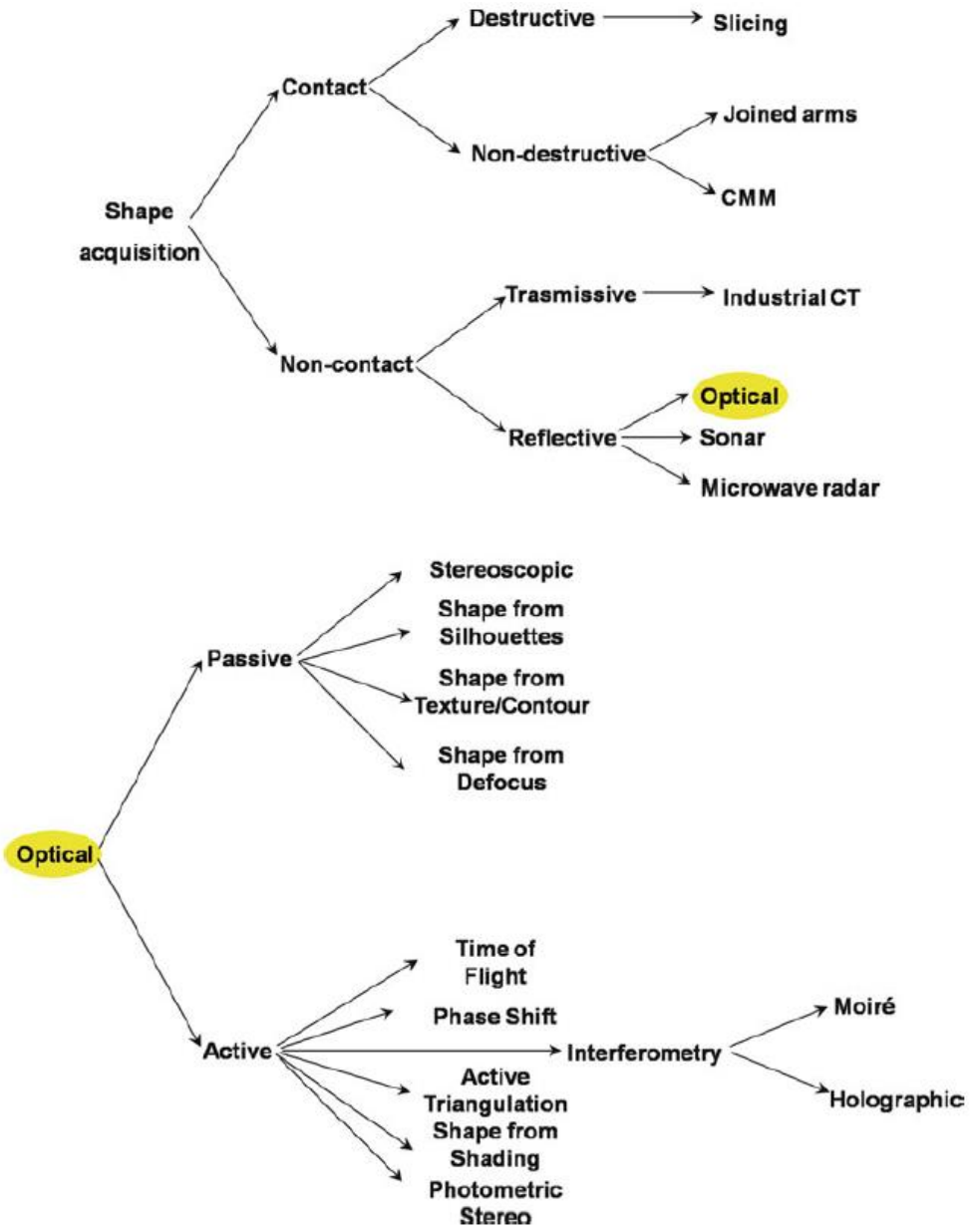
Závěrečná fáze celého procesu se odvíjí od účelu pořízení skenu. Model získaný za účelem inspekce rozměrů může mít jiné nároky na přesnost než model pořízený pro využití ve virtuální realitě. Optimalizace modelu s ohledem na jeho využití je důležitá pro dosažení maximální efektivity z hlediska využití paměti a výpočetního výkonu [1].

1.2 Principy 3D skenování

V přechozí kapitole byl představen obecný postup 3D skenování reálných objektů. Tato část se zaměřuje na principy získávání vstupních dat, respektive na principy fungování jednotlivých 3D skenerů.

Podle autorů knihy 3D Surface Reconstruction, Multi-Scale Hierarchical Approaches [1] mohou být jednotlivé skenery hodnoceny podle následujících kritérií: přesnost, rozlišení, rychlost, flexibilita použití, destruktivita vůči měřenému objektu, robustnost proti okolním podmínkám a pořizovací cena.

Z fyzikálního pohledu lze skenery rozdělit do dvou základních skupin. První z nich funguje na principu mechanického kontaktu s povrchem skenovaného objektu, a proto se souhrnně tyto skenery označují jako kontaktní. Druhá skupina skenerů využívá k měření geometrie elektromagnetické záření, nebo akustické vlny, které se odráží od povrchu snímaného objektu [1] [2]. Tyto skenery jsou označovány jako bezkontaktní. Celkový přehled klasifikace principů získávání 3D dat znázorňuje obrázek 1.4.



Obrázek 1.4 Přehled klasifikace metod 3D skenování [1]

Skupinu kontaktních skenerů můžeme dále rozdělit na skupinu přístrojů označovaných jako CMM[1] (viz obrázek 1.5) a na víceosá ramena s více stupni volnosti [1]. Princip detekce rozměrů objektu je však pro obě podskupiny stejný. Přístroj k měření používá kontaktní sondu, což ho předurčuje spíše pro měření menších předmětů. Podstatnou nevýhodou tohoto způsobu je časová náročnost metody, nebezpečí poničení měřeného objektu, nemožnost měřit předměty z měkkého materiálu a nízký počet změřených bodů. Další problém může nastat s předměty složitějších tvarů. Výše zmíněné nevýhody jsou však vyváženy vysokou přesností měření [1] [2] [5] [6].

[1] CMM – anglicky Coordinate Measuring Machines, česky Souřadnicové měřicí přístroje

Obrázek 1.5 Coordinate Measuring Machine (CMM) [7]

Z hlediska možností použití rozšířenější skupinu pak tvoří skenery bezkontaktní. Jejich využití můžeme nalézt např. v průmyslu, zdravotnictví a díky možnosti snímat i větší předměty také v architektuře [2] [8]. Tato práce se dále zaměřuje pouze na kategorii optických bezkontaktní skenerů.

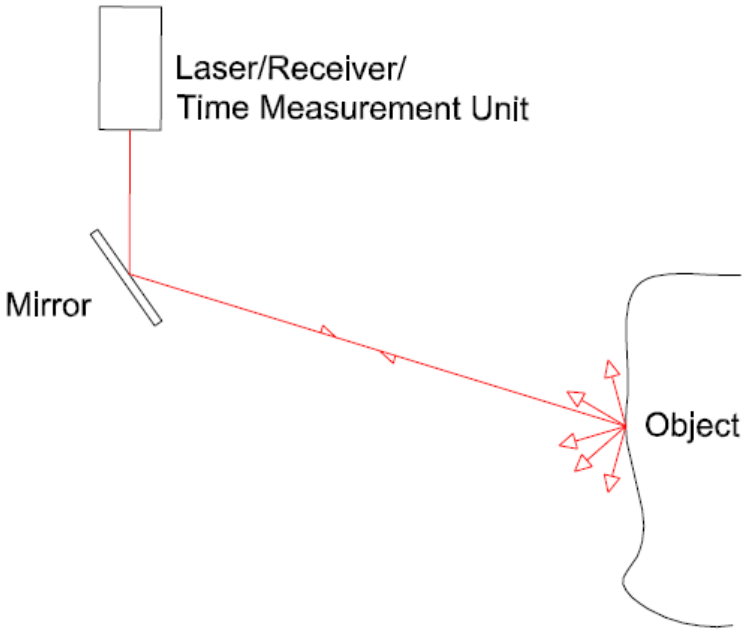
Mezi hlavní výhody skenerů využívající optický princip obecně patří rychlost měření (řádově desetitisíce až statisíce bodů / s), vysoké rozlišení a také možnost měřit i geometricky složité objekty [1]. Tyto parametry se však mohou pro jednotlivé optické metody lišit. Optické skenery můžeme rozdělit podle principu fungování na pasivní a aktivní [1] [2] [5].

Pasivní metody se vyznačují tím, že při procesu skenování zařízení neemituje žádné záření a pro měření je využíváno pouze záření okolního prostředí, které dopadá na skenovaný předmět [1] [2]. Běžně používanou pasivní metodou je stereoskopie, při které je tvar předmětu vyhodnocen pomocí dvou kamer, které využívají CCD snímače. V obrazech získaných z obou kamer jsou nalezeny korespondenční páry významných bodů (nejčastěji rohy nebo hrany objektů) a z jejich rozdílné polohy v rámci každého obrazu je následně pomocí triangulace a znalosti fyzického rozestavění kamer, ohnisek a rozlišení jednotlivých čipů možné dopočítat prostorové souřadnice významných bodů. Mezi přednosti stereoskopie patří nižší cena a odolnost vůči okolním podmínkám, avšak nalezení korespondenčních párů bývá z hlediska výpočetního výkonu náročné a v porovnání s ostatními metodami dosahují výsledky nižší přesnosti [1] [2].

Aktivní metody na rozdíl od pasivních používají vlastní zdroj záření. To dopadá na povrch měřeného objektu a na základě detekce odraženého záření je stanovena jeho geometrie. Mezi aktivní metody patří například princip měření času (obrázek 1.6), měření fázového posunu nebo princip triangulace (obrázek 1.7) [1] [2] [8]. Podle Boehlera

a Marbse [8] dosahuje metoda triangulace na kratší vzdálenosti větší přesnosti než další dvě zmíněné metody, ale chyba s rostoucí vzdáleností mezi měřeným objektem a skenerem roste parabolicky. U metody měření času a fázového posunu roste chyba se zvětšující se vzdáleností lineárně a jen mírně.

Skener fungující na principu měření času neboli ToF[2] vyšle laserový paprsek směrem k povrchu skenovaného předmětu a vyhodnocuje čas mezi vysláním a detekováním odraženého paprsku. Souřadnice měřeného bodu jsou stanoveny ze směru, kterým je daný paprsek vyslán a doby jeho letu [1] [8]. Z tohoto principu vyplývá, že je možné v daný okamžik měřit pouze jeden bod, avšak i přesto se rychlost snímaní pohybuje v rozsahu mezi 10 tisíci až 100 tisíci body za sekundu. Přesnost těchto skenerů se odvíjí od přesnosti směrování paprsku a měření času. Rozdíl vzdálenosti 1 mm odpovídá času v řádu pikosekund [1] [8].



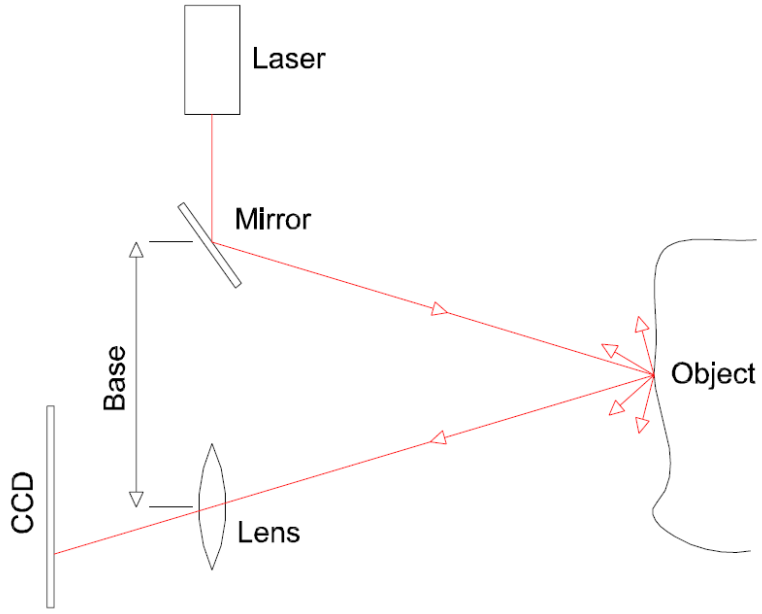
Obrázek 1.6 Schéma principu skeneru fungujícím na principu ToF [8]

Princip měření fáze je ve své podstatě také měřením času a v mnoha ohledech se podobá metodě ToF. Rozdíl však spočívá v tom, že se neměří časový interval mezi vyslaným a přijatým paprskem, ale vysílaný paprsek je modulován harmonickou funkcí a výsledná vzdálenost je vypočtena z fázového posunu mezi vyslaným a přijatým paprskem [1] [8]. U této metody je nutné předcházet problémům s detekováním fázového posunu mimo stejnou periodu. To lze zabezpečit například použitím signálů o více frekvencích. Metoda fázového posunu se i svými parametry velmi blíží metodě ToF, ale získává navrch v rychlosti snímání, která může dosahovat hodnot až 500 tisíc bodů/s [1].

[2] ToF – anglicky Time of Flight, česky Doba letu

Metoda aktivní triangulace spočívá v promítání laserového paprsku na povrch měřeného předmětu a jeho poloha je snímána kamerou používající CCD snímač. Stanovení souřadnic daného bodu probíhá pomocí triangulace – stejně jako tomu bylo u stereoskopie [1] [2] [8]. Promítaným vzorem však nemusí být pouze jeden bod, může jím být i pruh, mřížka nebo specifický optický kód. V takovém případě lze detekovat více bodů naráz, díky čemuž lze nasnímat velké množství bodů během krátkého času [1].

Obecným problémem optických metod však může být vliv okolního světla, materiálu a povrchu skenovaného předmětu. U lesklých nebo transparentních materiálů může docházet k nechtěným odrazům nebo pohlcením vysílaného světla a ke vzniku falešných bodů. Řešením může být nanesení tenké vrstvy matného spreje, což však není vždy možné a může také docházet k mírnému zkreslení snímaného povrchu [5].



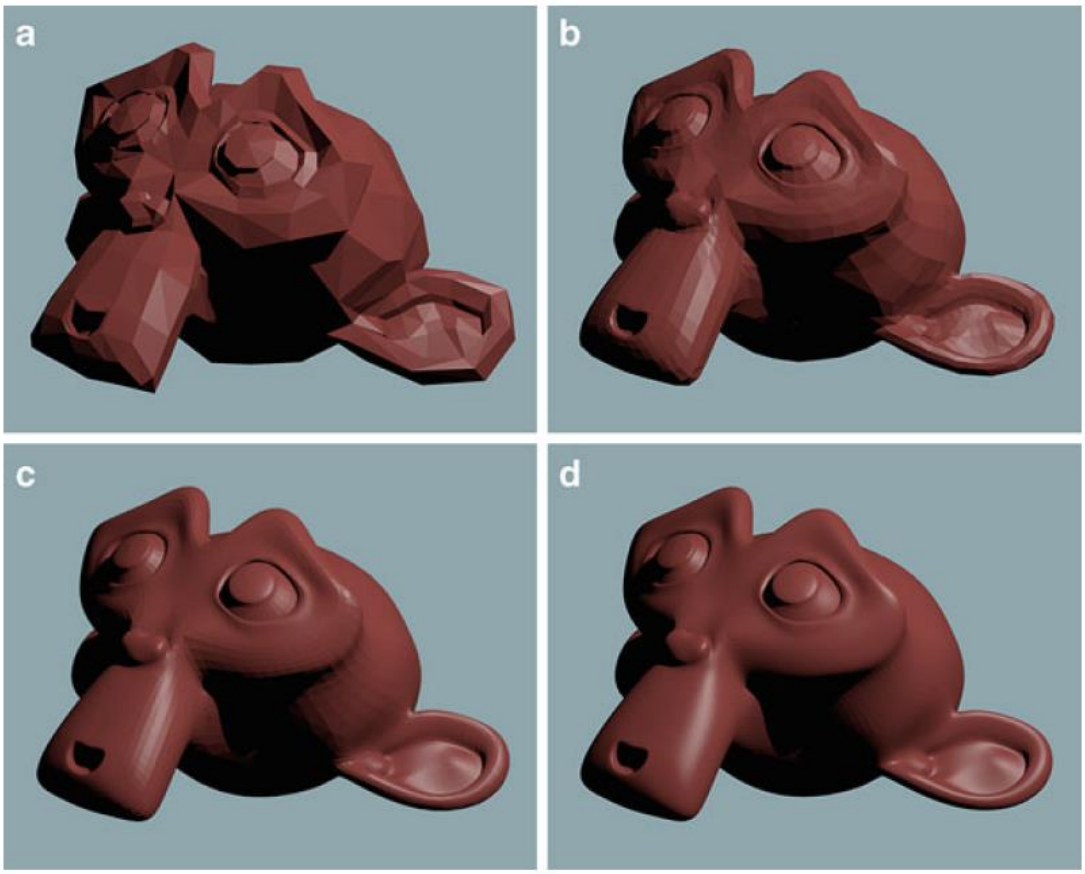
Obrázek 1.7 Schéma principu skeneru fungujícího na principu aktivní triangulace [8]

1.3 Typy 3D dat

Existuje více možností, jak mohou být data 3D modelů reprezentována. Zvolená varianta se může odvíjet od účelu použití daného modelu, nebo vycházet z principu vytvoření daných 3D dat. V této podkapitole budou stručně vysvětleny jednotlivé varianty.

Surová data pořízená 3D skenery jsou ve formě point cloudů. Jedná se o množinu bodů, které co nejpřesněji reprezentují tvar reálného objektu. Pokud by bylo potřeba získat představu o celém povrchu jakožto o souvislé ploše, využívá se aproximace pomocí polygonů, jejichž vrcholy jsou tvořeny body získanými ze 3D skeneru. Pro tuto

reprezentaci plochy se používá anglické označení mesh. Nejčastěji používaný typ meshe má trojúhelníkovou strukturu [9]. Aproximovaný povrch objektu je složen z trojúhelníků, které jsou definovány souřadnicemi jejich vrcholů a normálou roviny, ve které se daný trojúhelník nachází. Problém může nastat při modelování oblých objektů, kdy při velkém zvětšení zjistíme, že povrch není dokonale hladký [9]. Řešením může být rozdělení povrchu na menší polygony, čímž je zajištěno zmenšení odchylky mezi aproximovaným a ideálním povrchem [1]. Touto metodou však nelze docílit dokonale hladkých povrchů pro libovolně velké přiblížení. Metodu dělení polygonů znázorňuje obrázek 1.8.



Obrázek 1.8 Vliv dělení polygonů na zakřivené objekty. Vzorový model je tvořen trojúhelníkovými polygony. [1]

Pro dosažení dokonale hladkých oblých povrchů se nabízí použití metody známé pod zkratkou NURBS [1] [9]. Tato metoda spočívá v rozdělení povrchu na vážené kontrolní body, kterým se říká uzly. Povrch je následně vypočítán pomocí interpolace těchto kontrolních bodů, čímž je zajištěno hladké zobrazení povrchů při jakémkoliv zvětšení modelu [9].

Poslední variantou, které bude zmíněna v této práci, je způsob reprezentace CSG (z angličtiny Constructive Solid Geometry). Tento způsob reprezentace 3D dat spočívá ve vytváření složitějších tvarů pomocí základních geometrických tvarů jako je např. koule, kvádr a válec a booleovskými operacemi prováděnými nad nimi (sjednocení, průnik,

rozdíl) [1] [9]. Tato metoda je využívána především při vytváření modelů pomocí CAD nástrojů, avšak pro vytváření složitějších tvarů umí CAD nástroje pracovat i s polygonální reprezentací 3D dat [1].

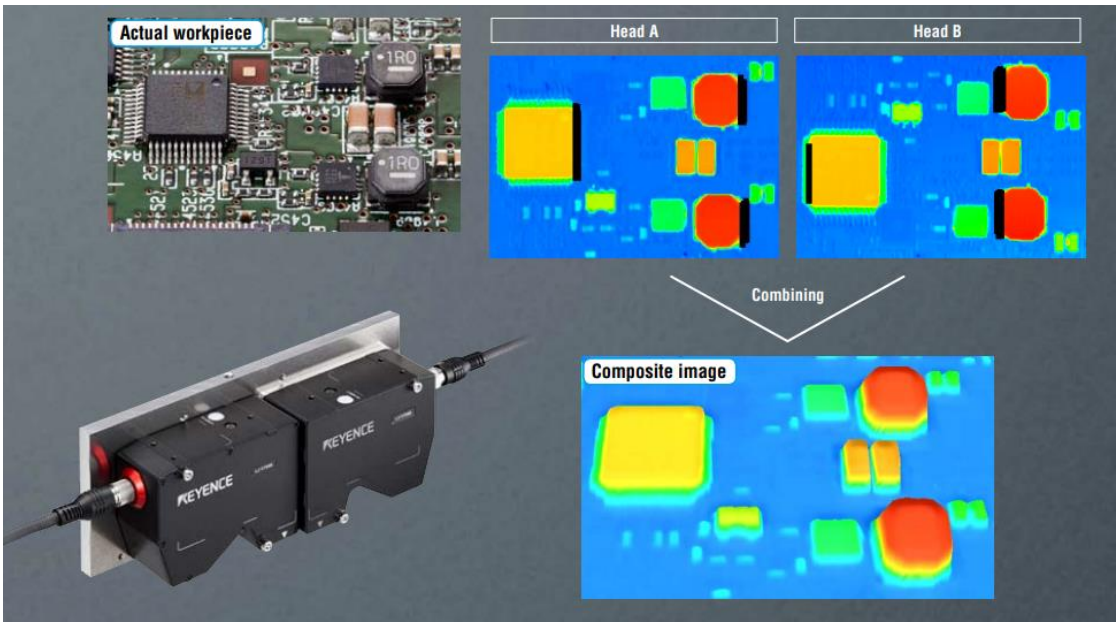
2. VOLBA VHODNÉHO HARDWARU

Tato kapitola se věnuje hardwarové koncepci skenovacího systému. Ten lze rozdělit na samotný 3D skener, periferní komponenty, které budou zajišťovat pohyb skeneru, nebo skenovaného dílu a na řídicí jednotku, která bude řídit celý proces skenování.

2.1 Výběr 3D skeneru

Z přehledu uvedeného v předchozí kapitole je patrné, že existuje poměrně široké spektrum možností 3D skenování. Profesionální skenery jsou však poměrně drahá zařízení a v současnosti firma Acam Solution žádným takovým zařízením nedisponuje. Tento fakt zužuje možný výběr na skenery, které mohou být zapůjčeny pro účely testování od výrobce skenerů Keyence. Skenerem, který bude využit pro účely této práce, je Keyence LJ-V7080 s řídicí jednotkou LJ-V7001P.

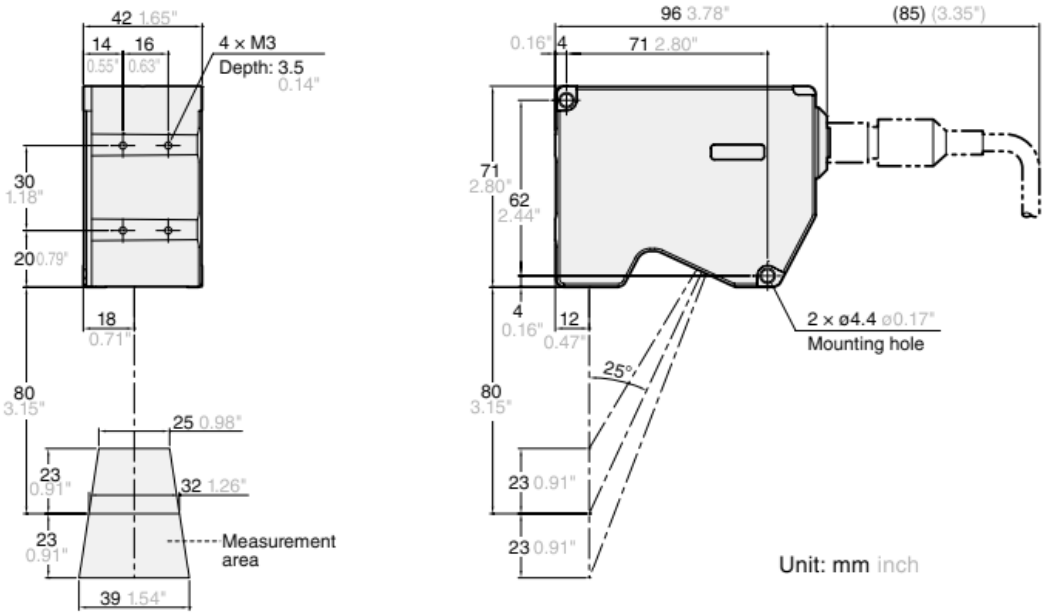
Řídicí jednotka LJ-V7001P zajišťuje ovládání a nastavení skeneru a také jeho komunikaci s nadřazenou řídicí jednotkou, kterou může být například PLC nebo PC. Nastavení skeneru je prováděno pomocí připojeného PC a programu LJ-Navigator2. Řídicí jednotka navíc umožňuje připojení dvou skenerů najednou, což umožňuje odstranění vlivu mrtvého úhlu, nebo při umístění obou hlav proti sobě pořízení skenu z obou stran předmětu najednou. Vliv mrtvého úhlu a jeho řešení pomocí dvou skenerů je vidět na obrázku 2.1. Zapůjčená testovací sada však obsahuje pouze jeden skener, takže tato funkce nemůže být použita.



Obrázek 2.1 Problematika a řešení mrtvého úhlu skeneru [10]

S ohledem na klasifikaci skenerů uvedené v kapitole 1.2 řadíme model LJ-V7080 mezi bezkontaktní optické skenery využívající princip aktivní triangulace, což znamená,

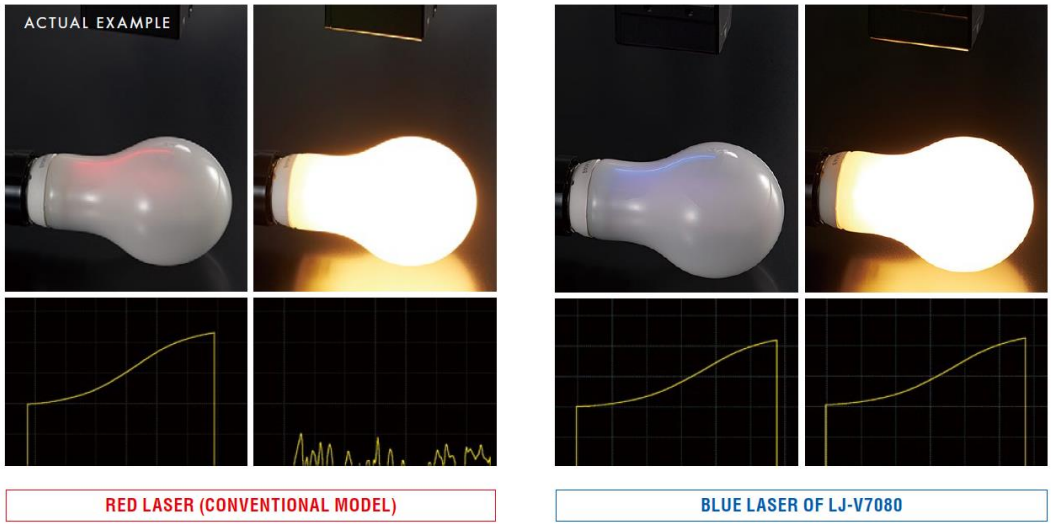
že skener vysílá světelný paprsek (v tomto případě ve tvaru pruhu) a snímač vyhodnocuje jeho odraz od povrchu snímaného objektu. Tato metoda skenování se jeví jako vhodná pro účely této práce. Jedním z hlavních parametrů skeneru, který má značný vliv na přípustné rozměry skenovaných dílů je referenční vzdálenost a rozsah oblasti měření. Referenční vzdálenost tohoto modelu je 80 mm a rozsah oblasti měření je 23 mm na každou stranu od referenční vzdálenosti. Maximální šířka profilu měřeného na bližší hranici rozsahu je 25 mm a na vzdálenější hranici je to 39 mm. Z toho vyplývá, že pomocí tohoto skeneru lze během jednoho skenovacího procesu získat data pouze o předmětu s rozměry maximálně několika desítek milimetrů. Pro skenování větších předmětů by bylo vhodné použít skenery s většími rozsahy, případně by šlo využít skládání výsledného skenu z více dílčích skenů. Se spojováním skenů do jednoho výsledného se však pro účely této práce nepočítá. Důvodem je především zvyšující se časová náročnost pořizování a následného zpracování takto skládaného skenu. Schéma rozměrů a měřicí oblasti skeneru LJ-V7080 je na obrázku 2.2. Celková specifikace skeneru je k dispozici v datovém listu, který se nachází v příloze této práce.



Obrázek 2.2 Schéma rozměrů a měřicí oblasti skeneru Keyence LJ-V7080 [10]

Jako zdroj světla skener využívá modrý laser o vlnové délce 405 nm, což oproti běžně používanému červenému laseru přináší několik výhod. Tou první je, že díky kratší vlnové délce je získaný obraz ostřejší [11]. Druhou výhodu představuje možnost měření předmětů o vysoké teplotě. To není při použití červeného laseru možné, jelikož předměty, které mají vysokou teplotu, emitují elektromagnetické záření, jehož vlnová délka se může

blížit vlnové délce červeného světla a mohlo by tak docházet ke zkreslení pořízených dat [11]. Vliv tohoto nežádoucího efektu je znázorněn na obrázku 2.3.



Obrázek 2.3 Porovnání použití červeného a modrého laseru pro 3D skenování teplo vyzařujících předmětů [11]

Mezi další přednosti těchto 3D skenerů patří využití dvojité polarizace zdrojového světla (X-polarizace a Y-polarizace). To umožňuje eliminovat odrazy vznikající na hranách předmětů, které by mohly být detekovány jako falešné body v prostoru. Odrazy vzniklé na hranách předmětu se projevují rozdílnou intenzitou X-polarizovaného a Y-polarizovaného dopadajícího světla. Díky tomuto rozdílu mohou být taková data vyhodnocena jako chybová a nejsou brána v potaz [11].

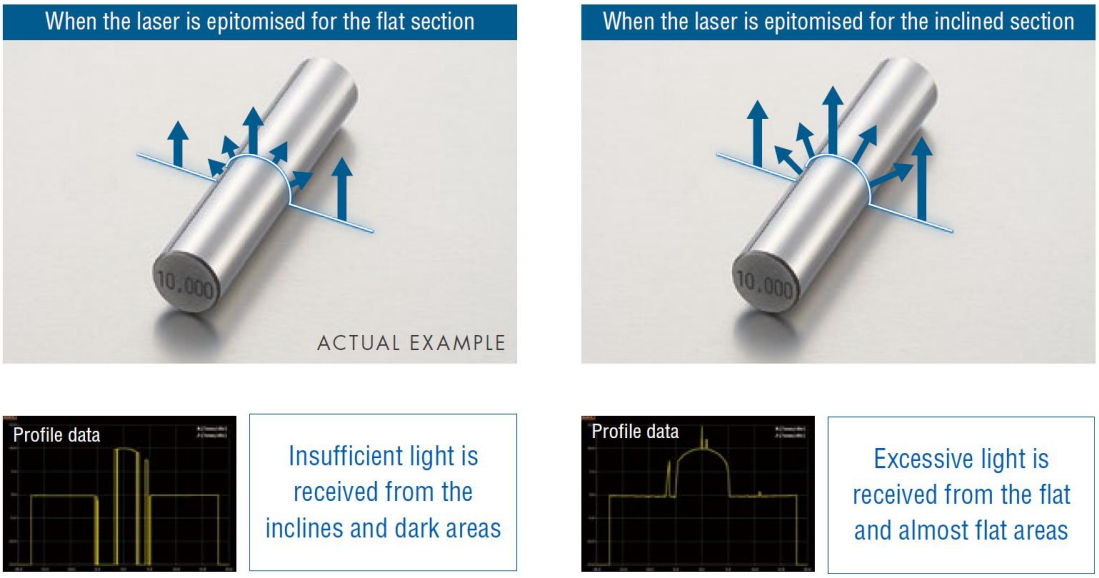
Skenery řady LJ-V7000, mezi které patří i zde použitý model, se spoléhají na procesor GP64 a čip HSE^3-CMOS. Díky těmto dvěma komponentám zvládá skener pořizovat snímky s rychlostí až 64 kHz, což odpovídá expozičnímu času 15,6 μs [11]. Takto rychlá snímací frekvence a závěrka je vhodná pro detekci rychle se pohybujících předmětů. Pro tuto práci může být vysoká snímací frekvence přínosná z hlediska rychlosti celého procesu skenování. Snímací frekvence 64kHz je však nejvyšší možná a různé pomocné funkce zpracování dat (např. eliminace odrazů na hranách pomocí dvojité polarizace) mohou snímací frekvenci skeneru snižovat.

Pro srozumitelnost celého procesu je vhodné poznamenat, že jeden snímek zachycuje pouze jeden profil skenovaného dílu, nikoliv celý předmět naráz. Model LJ-V7080 pořídí naráz v jednom profilu 800 bodů, přičemž poloha bodu v rámci daného profilu slouží k výpočtu souřadnice v ose x a souřadnice v ose z je dána změřenou výškou předmětu odpovídající danému bodu profilu. Souřadnice bodu v ose y je dopočítávána z pořadí profilu v rámci celého procesu skenování, použité vzorkovací frekvence a rychlosti, jakou se pohybuje skener vůči skenovanému dílu. Výpočet souřadnice osy y je proveden podle rovnice

$$y = i_{profil} \cdot \frac{v}{f}, \tag{2.1}$$

kde y je souřadnice v ose y [m], i_{profil} je pořadí profilu v rámci procesu skenování, v [m/s] je rychlost pohybu skeneru vůči předmětu a f [Hz] je frekvence snímaní. Alternativou pro určení souřadnice osy y je použití vstupu z enkodéru, jehož signál je přiveden na odpovídající svorky řídicí jednotky skeneru.

Výše zmíněný čip také poskytuje vysoký dynamický rozsah, což je parametr, který určuje, jak moc světlé a tmavé oblasti mohou být snímačem zachyceny naráz do jednoho snímku. Čipy s nízkým dynamickým rozsahem mohou mít problém s předměty, které disponují plochami s vysokou a zároveň nízkou odrazivostí, což může vést k nežádoucímu zkreslení získaných dat [11]. Tato problematika je znázorněna na obrázku 2.4.



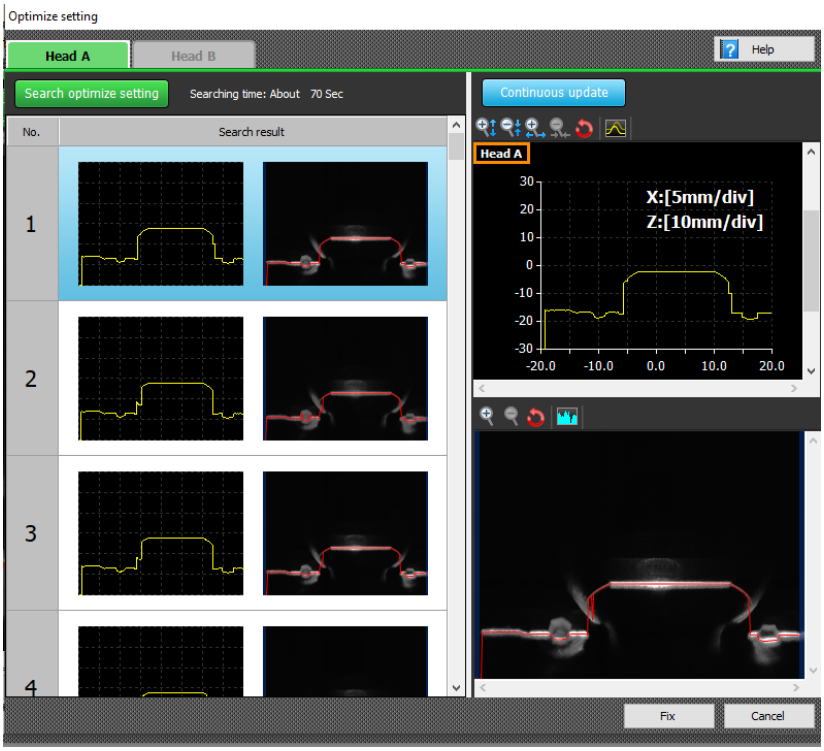
Obrázek 2.4 Chyby vzniklé kvůli nízkému dynamickému rozsahu čipu [11]

2.1.1 Nastavení 3D skeneru

Jak již bylo zmíněno výše, nastavení parametrů skeneru probíhá pomocí programu LJ-Navigator2, který je spuštěn na PC, které je připojeno k řídicí jednotce skeneru pomocí USB nebo ethernetového rozhraní. Program zobrazuje živý náhled snímaných dat, díky čemuž má uživatel okamžitou zpětnou vazbu, což značně usnadňuje proces ladění parametrů skeneru. Zobrazovanými daty jsou výškový profil a obraz snímaný kamerou. Jedním z parametrů, který je potřeba nastavit, je režim spouště neboli triggeru. První možností je režim *continuous*, při kterém je jako zdroj spouště využit vnitřní hodinový signál kontroléru. Dalšími režimy jsou *external trigger*, kdy využíváme externí zdroj signálu a *encoder trigger*, který využívá signál pocházející přímo z enkodéru. Dalším

podstatným parametrem je frekvence snímání. Jak již bylo zmíněno výše, maximální snímací frekvence je 64 kHz, avšak reálná maximální snímací frekvence se odvíjí od nastavení ostatních parametrů skeneru. Skener umožňuje provádět měření i v dávkovém režimu, kdy je od spuštění pořízen zadaný počet profilů, který je následně uložen jako jeden soubor. Maximální počet profilů pořízených v rámci jedné dávky je 15000. Posledním zde zmíněným nastavitelným parametrem je expoziční čas kamery. Uživatel může nastavit jednu z 10 hodnot, které se nachází v intervalu od 15µs do 10ms. Expoziční čas je vhodné volit s ohledem na rychlost pohybu skenovaného předmětu (resp. skeneru) tak, aby nedocházelo k pohybové neostrosti pořízených snímků.

Pro nastavení většiny parametrů je vhodné použít nástroj automatické optimalizace parametrů, který pořídí několik vzorků s různě nastavenými parametry a uživatel pak vybere vzorek, jehož profil nejlépe odpovídá skenovanému předmětu (viz obrázek 2.5). Po vybrání vzorku dojde k nastavení odpovídajících parametrů. Tento postup byl využit při pořizování testovacích dat pro tuto práci.



Obrázek 2.5 Ladění parametrů skeneru pomocí programu LJ-Navigator2

Na obrázku 2.6 vidíme testovací díl, jehož rozměry byly měřeny při testovacím měření. Vzhledem k tomu, že na hranách testovacího dílu docházelo k nechtěným odleskům, bylo zapotřebí použití funkce polarizace zdrojového světla, což je jeden z faktorů, který má vliv na snížení maximální snímací frekvence. Poměrně vysoká odrazivost povrchu dílu vedla rovněž k využití režimu vysokého dynamického rozsahu. Pro potlačení šumu v rámci jednoho profilu byl použit mediánový filtr s velikostí 3. Režim spouště byl nastaven na možnost *continuous* a maximální možná snímací

frekvence při daném nastavení byla 1 kHz. Při testování byl využit dávkový režim snímání. Výsledné nastavení skeneru použité při testování je uvedeno v tabulce 2.1.

Obrázek 2.6 Testovací konstrukční díl

Tabulka 2.1 Nastavené parametry 3D skeneru LJ-V7080

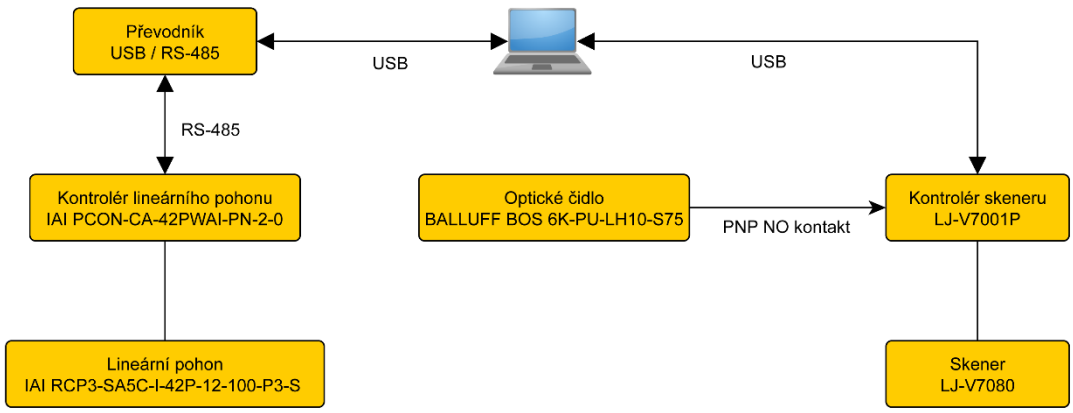
Nastavitelný parametr	Hodnota parametru
Trigger mode	Continuous
Sampling frequency	1 kHz
Pitch between triggers	Do not set
Batch measurement	ON
Batch profiles	4000
CMOS sensitivity	High dynamic range 2
Exposure time	960 µs
Peak detection sensitivity	5
Interpolations points invalid data	8
Peak selection	Standard
Peak width filter	OFF
Imaging mode	Multi emission
Emission times	3 times
Filter process X-axis smoothing	4
Filter process X-axis median	3
Filter process time-axis averaging	1
Filter process time-axis median	OFF

2.2 Výběr periferií

Periferiemi se rozumí všechna zařízení kromě 3D skeneru, která byla použita při realizaci skenovacího systému. Tím hlavním je akční člen, který bude zajišťovat lineární

pohyb skeneru, nebo skenovaného předmětu během skenování. S ohledem na náročnost realizace bylo rozhodnuto upevnit skener na statický úhlový držák a pohybovat skenovaným předmětem. Vzhledem k malým maximálním rozměrům (omezeno šířkou profilu skeneru) a tím pádem i malým hmotnostem skenovaných dílů, nemusíme na tento fakt při vybírání pohonu brát přílišný ohled. Prakticky jedinými požadavky, které při výběru pohonu musíme splnit, jsou dostatečně velký rozsah pohybu a schopnost pohonu pohybovat se konstantní rychlostí bez velkých odchylek, protože z rychlosti pohybu je dopočítávána souřadnice bodů v ose y. Na druhý požadavek by nebylo nutné brát ohled v případě použití enkodéru, jehož signál by mohl být připojen na vstup řídicí jednotky skeneru, avšak enkodér nebyl při testování k dispozici. Pro realizaci skenovacího systému byl k dispozici lineární pohon IAI Corporation RCP3-SA5C-I-42P-12-100-P3-S s řídicí jednotkou PCON-CA-42PWAI-PN-2-0. Tento pohon s rozsahem pohybu 100 mm splňuje výše zmíněné požadavky. Jedinou nevýhodou tohoto lineárního pohonu je jeho nejnižší rychlost pohybu, při které může být zaručena plynulost pohybu. Při rychlostech pod 16,8 mm/s by totiž mohlo docházet k vibracím a nerovnoměrnému pohybu. Kontrolér lze propojit s PC pomocí rozhraní RS-485 a pomocí programu PC Interface Software for RC/EC od firmy IAI Corporation lze nastavit až 64 pozic a jim odpovídajících rychlostí, do kterých se může pohon pohybovat. Pro účely této práce stačí nastavit pouze počáteční a koncový bod pohybu.

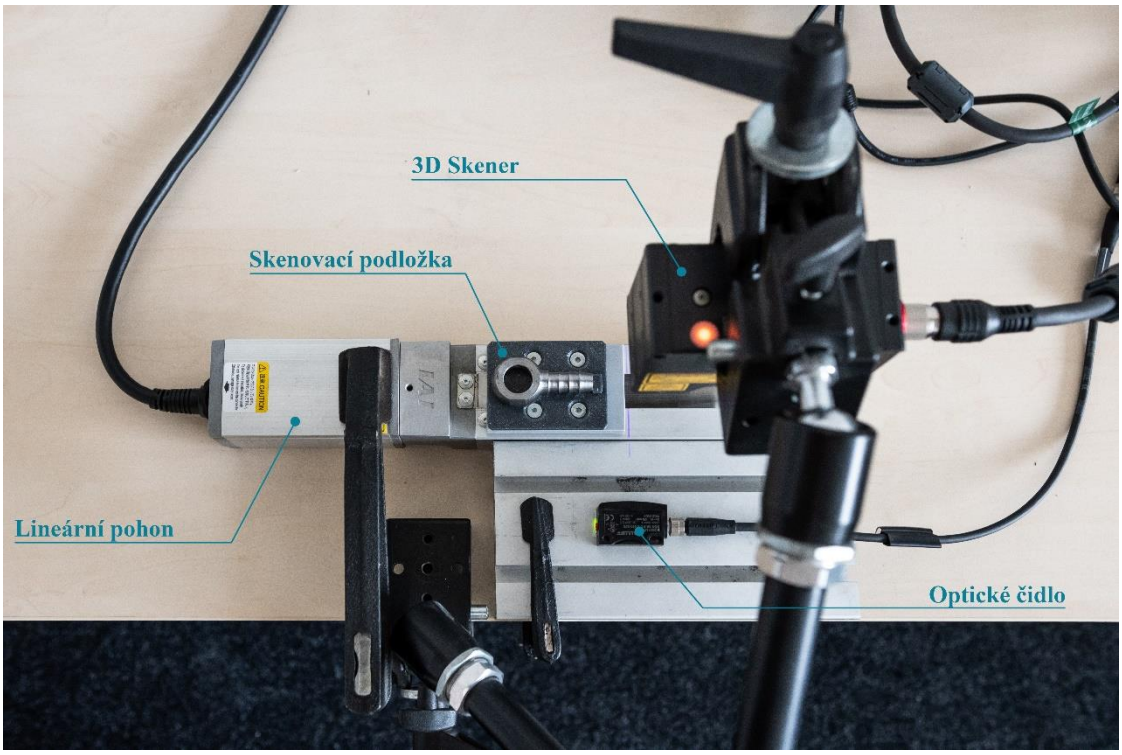
Pro synchronizaci zahájení pohybu skenovaného dílu a spuštění skenovacího procesu (start dávky) bylo použito optické čidlo s potlačeným pozadím značky BALLUFF BOS 6K-PU-LH10-S75. Výstup čidla je přiveden na odpovídající vstup řídicí jednotky LJ-V7001P. Řídicí jednotka skeneru, skener, kontrolér lineárního pohonu, lineární pohon a optické čidlo jsou napájeny pomocí zdroje Weidmüller PRO ECO 240W 24V 10A. Schéma zapojení použitých zařízení ukazuje na obrázek 2.7.



Obrázek 2.7 Schéma zapojení testovacího pracoviště

2.3 Skenovací sestava

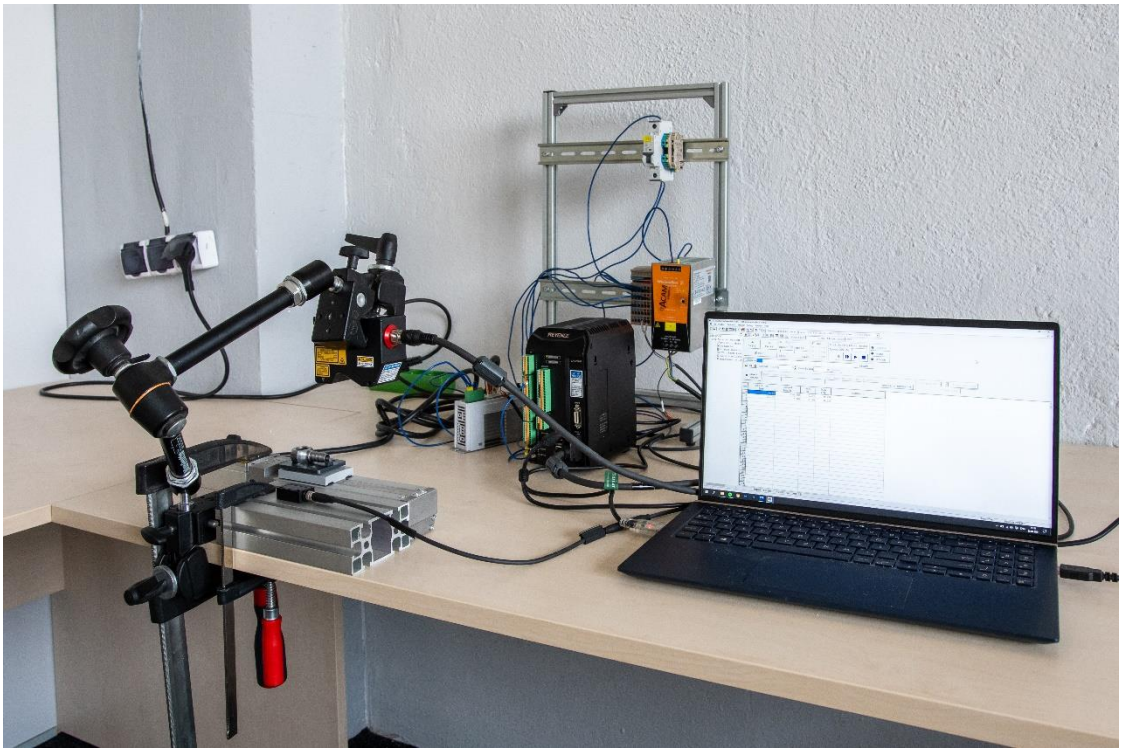
Pro snadnější popis sestavy je definován ortogonální souřadnicový systém tvořený osami x, y a z. Ve vodorovné rovině tvořené osami x a y se nachází skenovací podložka, která je připevněna na pojezd lineárního pohonu, který zajišťuje pohyb skenovaného dílu během procesu. Tato podložka byla vyrobena na 3D tiskárně a její tvar pomáhá aretovat díl do požadované polohy. Skener se nachází nad rovinou x-y, kde je staticky upevněn pomocí nastavitelného úhlového držáku a jeho měřicí ústrojí směřuje dolů kolmo ke skenovací podložce, přičemž promítaný pruh světla je rovnoběžný s osou x. Měřicí ústrojí skeneru se nachází ve výšce 94,5 mm od skenovací podložky. Celá sestava je upevněna pomocí svěráků k desce stolu, tak aby nemohlo dojít během skenovacího procesu k nechtěným pohybům skeneru a skenovaného dílu. Hlavní funkční část skenovací sestavy je zobrazena na následujícím obrázku 2.8.



Obrázek 2.8 Uspořádání skenovací sestavy

Celá sestava je řízena z notebooku, který je připojen k řídicí jednotce skeneru a lineárního posuvu (viz obrázek 2.9.) Na notebooku také běží program sloužící k automatickému měření geometrických rozměrů skenovaného dílu, který byl vytvořen v rámci této práce. Proces skenování začíná spuštěním lineárního posuvu, na kterém je již připravený díl, jehož rozměry chceme měřit. Spuštění motoru probíhá z programu PC Interface Software for RC/EC, kdy po stisknutí tlačítka přejede posuv z počáteční do koncové pozice, které byly do kontroléru motoru předem nahrány. Spuštění samotného skenování zajišťuje optické čidlo detekující pohyb pojezdu, které je připojeno na

odpovídající vstup řídicí jednotky skeneru V7001P. Skener následně pořídí 4000 profilů se snímací frekvencí 1 kHz a získaná data jsou odeslána a uložena do notebooku. Snímací frekvence je volena jakožto nejvyšší možná pro dané nastavení skeneru. Počet profilů pořízených během jedné dávky vychází z nejnižší možné rychlosti, kterou se může pojezd pohybovat (16 mm/s), rozměru skenovaného dílu ve směru skenování (necelých 50 mm) a zvolené snímací frekvence (zde 1 kHz). Pokud by se pojezd mohl pohybovat pomaleji, nebo pokud by mohla být nastavena vyšší snímací frekvence, bylo by z hlediska přesnosti měření vhodné pořídit co nejvíce profilů. Pořízená data jsou zpracována navrženým programem (viz kapitola 4) a výsledné změřené hodnoty jsou vytištěny do konzole a uloženy do výstupního souboru ve formátu CSV. Popsaný systém slouží pouze jako demonstrativní ukázka řešení zadané problematiky. V případě reálného využití by bylo možné jednotlivé komponenty ovládat například pomocí nadřazeného programu běžícímu na PLC, přičemž výstupy programu by byly přizpůsobeny požadovanému formátu.



Obrázek 2.9 Testovací pracoviště

3. VÝBĚR VHODNÉHO SOFTWAROVÉHO NÁSTROJE

V této kapitole jsou shrnuty výsledky rešerše týkající se výběru vhodného softwaru pro práci s 3D daty. Jednotlivé softwarové nástroje jsou posuzovány optikou tématu této diplomové práce tzn. jejich využitím při automatizovaném procesu skenování vyrobených dílů a následné kontroly jejich rozměrů oproti referenčním hodnotám. Po konzultaci s firmou byl výběr zúžen na následující softwarové nástroje: PCL, GOM Inspect, Geomagic Control X od společnosti 3D systems a Halcon.

3.1 Point Cloud Library

Point Cloud Library, neboli zkráceně PCL, je bezplatná volně dostupná knihovna pro práci s 2D a 3D daty vydávaná pod licencí BSD, díky čemuž může být použita jak pro výzkumné tak i pro komerční účely [12]. Je psána v jazyce C++ a může být použita na platformách Windows, Linux, MacOS a Android. Knihovna obsahuje nejrůznější algoritmy pro zpracování 3D dat ve formě point cloudů, které jsou rozděleny do menších dílčích knihoven, což napomáhá využití PCL i na zařízeních, která jsou více omezena svým výpočetním výkonem a pamětí [12].

Jednotlivé funkce PCL knihovny lze rozdělit do následujících kategorií: Filters, Features, Keypoints, Registration, KdTree, Octree, Segmentation, Sample Consensus, Surface, Range Image, I/O a Visualisation [13]. Není předpokládáno využití všech dílčích částí PCL, proto jsou zde stručně popsány jen ty, které by mohly být užitečné vzhledem k tématu této práce.

I/O – tato knihovna obsahuje funkce nutné pro čtení a zápis souborů obsahujících point cloudová data. Nabízí však i funkce pro práci s některými snímacími zařízeními [13].

Filters – většina 3D dat získaných skenováním je zatížena chybou v podobě falešných bodů, které neodpovídají realitě. To může být způsobeno jevy, které jsou uvedeny výše na konci kapitoly 1.2. Tyto body je potřeba odstranit, aby později nedocházelo k nežádoucímu ovlivňování procesu – např. při vyhodnocování lokálních vlastností point cloudu. Některé z těchto chybových bodů mohou být odstraněny na základě statistické analýzy okolí každého z těchto bodů [13].

Keypoints – keypoints neboli významné body jsou stabilní, neměnné body v obraze, jejichž kombinace s lokálními vlastnostmi těchto bodů poskytuje dostatečně přesnou představu o objektech zachycených v point cloudu. Významnými body objektů jsou typicky jejich rohy a hrany. Mezi metody hledání významných bodů nabízené touto knihovnou patří například Harrisův detektor, detektor AGAST, NARF, SIFT a další [13].

Features – tato část obsahuje mechanismy pro určování 3D vlastností jednoho konkrétního bodu, které jsou stanoveny na základě informací získaných z bodů, které leží v *k*-okolí zkoumaného bodu. Posuzované okolí bodu lze definovat jakožto počet

nejbližších bodů ležících v okolí zkoumaného bodu nebo jako množinu bodů, které se nacházejí uvnitř pomyslné koule o poloměru *r* se středem v zájmovém bodě. Pro efektivnější určení toho, které ze sousedních bodů zájmového bodu spadají do posuzovaného okolí, lze využít rozdělení původního modelu na menší oblasti pomocí funkcí z částí *KdTree* nebo *Octree*. Hledání nejbližších sousedních bodů pak probíhá pouze uvnitř této menší oblasti [13].

Registration – v této knihovně jsou k dispozici funkce, pomocí kterých lze zarovnat různé point cloudy vůči sobě. To lze využít při skládání rozsáhlé scény z více dílčích modelů nebo pro zarovnání dvou stejných point cloudů vůči sobě pro následné porovnání jejich geometrických vlastností. Při procesu registrace je hledána taková transformace, aby odchylka mezi referenčním a zarovnávaným point cloudem byla nižší než nastavená hodnota prahu. Algoritmus může však skončit i dřív, pokud je dosaženo maximálního počtu iterací stanovených uživatelem [13].

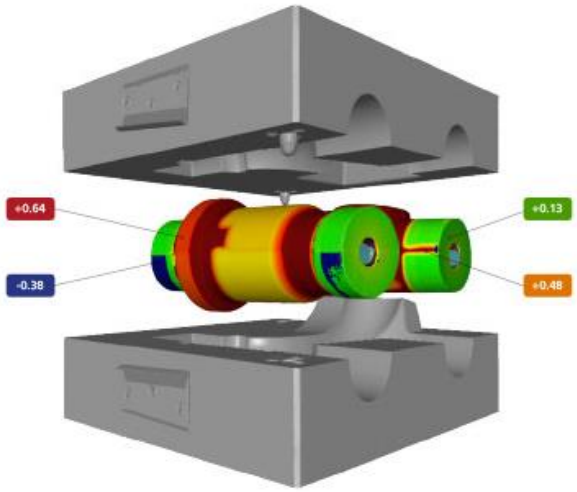
Vzhledem k množství implementovaných funkcí a k volně dostupné licenci knihovny se jedná o vhodný nástroj, který by mohl být využit pro realizaci automatického systému kontroly rozměrů konstrukčních dílů.

3.2 GOM Inspect

Program GOM Inspect je profesionální software sloužící ke zpracovávání 3D dat získaných nejrůznějšími metodami. Pro tuto práci je přínosné, že zvládá zpracovat takřka jakákoliv 3D data. Z nabízených implementovaných funkcí je zajímavá možnost práce s CAD modelem, který může sloužit jako referenční vzor pro naskenovaná data [14].

Aby mohl být naskenovaný model porovnán s tím referenčním, musí být tyto modely vůči sobě nejprve zarovnány. Program nabízí různé možnosti automatického zarovnání – lze zarovnat celý model s ohledem na co nejmenší odchylku v celém objemu, avšak v případě potřeby je možné provést zarovnání jen podle konkrétní části objektu. To se hodí v případě, kdy se kontrolovaný předmět skládá ze dvou nebo i více dílčích částí a je třeba zjistit správnost rozměrů celého předmětu. Může totiž nastat situace, kdy rovina jedné části předmětu vykazuje značné odchylky od celkového objektu a není jasné, jestli je vzniklá odchylka dána vadami jednotlivých dílů sestavy, nebo jsou díly vyrobeny správně a zjištěná odchylka vychází z jejich špatného sestavení [14].

Program umožňuje měřit vložený objekt nejen z pohledu jeho rozměrů, ale také z pohledu dalších geometrických vlastností, jakými jsou například kolmost, válcovitost nebo rovinatost. Zajímavá je i možnost geometrické analýzy dílů, které mají pasovat do sebe. Program dokáže automaticky stanovit například nejmenší průměr vnějšího a největší průměr vnitřního dílu a následně stanovit, jestli jsou tyto součástky kompatibilní či nikoliv [14]. Grafické znázornění výsledku této funkce ukazuje obrázek 3.1.



Obrázek 3.1 GOM Inspect - Inspekce kompatibility dvou dílů [14]

Veškerá analytická data lze zobrazovat přímo na povrchu modelu pomocí barevného spektra, samozřejmostí je i možnost vyčítání konkrétních diskrétních hodnot. Lze například stanovit i tolerance odchylek rozměrů a program automaticky vyhodnotí, jestli je díl vyroben v rámci tolerance či nikoliv [14].

Program GOM Inspect nabízí i nástroj pro obsluhu vlastní automatizované soustavy, kdy je skener umístěný na robotickém šestiosém rameni. V programu lze v grafickém prostředí naplánovat trajektorii robotického ramena a pořizování 3D skenů v konkrétních bodech trajektorie. Proces zpracovávání a vyhodnocování 3D lze do jisté míry automatizovat i pomocí skriptů, ale tyto výše zmíněné funkce jsou dostupné pouze v placené verzi programu [14].

Z hlediska možností, které program nabízí se jedná o velmi zajímavé řešení kontroly rozměrů a tolerancí konstrukčních dílů. V bezplatné verzi však program nenabízí možnost automatizování procesu zpracování 3D dat. S ohledem na tento fakt nebude tento program využit jakožto SW nástroj pro realizaci této práce. Mohl by však posloužit jako zdroj reference pro porovnání výsledků získaných pomocí PCL

3.3 Geomagic Control X

Tento program z dílny 3D systems je jednou z alternativ, které trh nabízí ke zpracování a vyhodnocování 3D dat. Co se týče možností využití tohoto programu, které výrobce sám prezentuje na svém webu, tak se prakticky moc neliší v tom, co nabízí již výše zmíněný GOM Inspect. Geomagic Control X sice oproti svému konkurentu postrádá například možnost programování trajektorie šestiosých robotických ramen, na kterých je uchycený 3D skener, ale co se týče samotného zpracování dat, tak se zdá, že nabízí stejné možnosti.

Samozřejmostí je možnost porovnat pořízená data s referenčním modelem, což nejčastěji bývá CAD model, podle kterého je měřený díl vyroben. I zde zarovnání referenčního a měřeného modelu a následné grafické znázornění odchylek pomocí barevné škály funguje jednoduše za využití implementovaných funkcí. Nechybí však ani možnost vyčítání konkrétních hodnot (ať už hodnot odchylek nebo například zobrazení kót mezi vybranými částmi). Modely lze zarovnat s ohledem na nejmenší celkovou odchylku, ale také jen podle konkrétní části modelu [15]. K této možnosti výrobce uvádí příklad využití, kdy je potřeba vůči sobě zarovnat poničenou (například ohnutou) součástku a její referenční model bez poškození.

Referenční model může být poskládán z více dílčích skenů. Naskenovaná 3D data lze také segmentovat, což lze využít například při oddělení modelu od jeho okolí (např. podložky, na které byl během skenování umístěn). Celý proces je možné do jisté míry automatizovat pomocí skriptování, které se provádí přes uživatelské rozhraní programu [15].

Program Geomagic Control X zcela jistě nabízí zajímavé řešení, co se týče zpracování 3D dat. Nevýhodou, kterou má společnou se svým konkurentem GOM Inspect, je, že není dostupný pod bezplatnou licencí v takovém rozsahu, aby byl vhodný pro cíl této práce. Zkušební bezplatná verze tohoto programu je dostupná jen po dobu 15 dní, což je méně, než nabízí GOM Inspect. Proto se využití programu Geomagic Control X nepředpokládá ani v roli referenčního nástroje pro porovnání měření.

3.4 Halcon

Program Halcon je dostupný pro operační systémy Windows, LinuX, MacOS ale i pro platformy postavené na čipu Arm [16]. To z něj v kombinaci s možností generování kódu do několika programovacích jazyků dělá široce využitelný nástroj pro práci v oblasti strojového vidění. Program disponuje vlastním vývojovým prostředím HDevEngine, které slouží pro vytváření programů ke zpracování obrazu. Poté lze kód přeložit do jazyka C, C++, C# nebo Visual Basic, aby byl co nejsnadněji implementovatelný do jakéhokoliv jiného projektu. HDevEngine obsahuje také běžné funkce pro získávání dat, kalibraci kamery nebo srovnávání tvarů [16].

Pro tuto práci má význam obzvlášť část nesoucí pojmenování 3D Vision, která obsahuje nástroje pro práci s 3D daty. Mezi nabízené funkce patří registrace, rekonstrukce, hledání objektů na základě shody povrchu, funkce pro inspekci povrchu a další [16].

Program také nabízí I/O rozhraní, jehož součástí jsou nástroje pro ovládání velkého množství zařízení třetích stran, které se běžně využívají pro pořizování obrazu a 3D dat.

Softwarový nástroj Halcon představuje další řešení v oblasti zpracování 3D dat, ale řadí se do skupiny k výše zmíněným programům GOM Inspect a Geomatic Control X – nenabízí volně dostupnou bezplatnou licenci, která by umožňovala splnit cíle této práce, tudíž se s jeho využitím nepočítá.

4. NÁVRH SW ŘEŠENÍ

Na základě rešerše, jejíž výsledky byly shrnuty v přechozí kapitole, bylo rozhodnuto realizovat tento projekt za využití knihovny funkcí s názvem Point Cloud Library. Hlavním faktorem, který vedl k tomuto rozhodnutí, bylo, že knihovnu PCL lze použít zcela bezplatně. Oproti ostatním zvažovaným variantám umožňuje rovněž nejjednodušší realizaci automatického zpracovávání vstupních dat a formát výstupních dat lze zcela přizpůsobit potřebám nadřazeného systému.

Knihovna PCL je volně dostupná ke stažení například na webu github.com. Vzhledem k tomu, že funkce knihovny využívají knihovny třetích stran (např.: Boost, Eigen, FLANN, VTK, …), byla instalace knihovny realizována pomocí open-source správce balíčků vcpkg od společnosti Microsoft. Vcpkg manager nainstaloval poslední dostupnou verzi PCL 1.11.1., ale při ověřování funkčnosti nainstalované verze PCL knihovny se objevil problém v podobě několika chybějících dílčích částí. Nainstalovaná verze postrádá části Visualisation a některé soubory z části Surface. Druhou variantou instalace, která byla vyzkoušena, bylo stažení instalačního souboru, který nainstaloval celou PCL knihovnu i s knihovnami třetích stran, avšak výsledek této instalace problém nevyřešil. V průběhu práce na tomto projektu se nepodařilo zprovoznit verzi PCL knihovny, která by obsahovala všechny funkční části. Nicméně chybějící část Visualisation má pouze zobrazovací funkci, takže její absence není stěžejní pro návrh mého programu. Pro zobrazování průběžných výsledků během vývoje programu byl využit volně dostupný program CloudCompare. Chybějící knihovny z části Surface mohou sice způsobit menší komplikace v podobě omezení některých funkcí při vývoji programu, tyto funkce však nejsou stěžejní a lze se bez nich obejít. Z těchto důvodů nebyl problém chybějících částí dále řešen a návrh programu proběhl s verzí knihovny nainstalované pomocí vcpkg manageru.

4.1 Koncept programu

V této kapitole bude představen rámcový koncept programu, který bude zajišťovat zpracování naskenovaných dat. Konkrétní části budou podrobněji rozebrány v následujících kapitolách.

Program bude sloužit k automatickému měření rozměrů naskenovaného konstrukčního dílu v segmentech stanovených uživatelem. Vstupy do programu jsou sken konstrukčního dílu a jeho CAD model. Model konstrukčního dílu má funkci reference v prostoru, podle které je sken dílu zarovnán. Zarovnání je nezbytné pro další fázi programu, ve které dochází k segmentaci jednotlivých části skenu. Části, které se mají segmentovat volí uživatel pomocí intervalů souřadnic – minimální a maximální hodnota v osách x, y, z. Intervaly jsou voleny podle modelu konstrukčního dílu, a proto je nezbytné sken před segmentací nejprve zarovnat vůči modelu. Tento proces zajišťuje

robustnost programu, protože nezáleží na umístění konstrukčního dílu při jeho skenování. Segmenty mohou obsahovat rovinné plochy nebo válcovité části skenu. Vzhledem k nespojité reprezentaci konstrukčního dílu ve formě jednotlivých bodů point cloudu, by bylo měření rozměrů složité. Proto chceme nalézt náhradní model segmentované částí, který co nejlépe aproximuje všechny body odpovídající danému segmentu. Jakmile jsou vypočteny náhradní modely všech segmentů, může být přistoupeno k samotnému měření. Program umí změřit vzdálenost mezi dvěma segmenty (přednostně mezi dvěma rovinami), úhel mezi dvěma segmenty (porovnávány jsou orientace náhradních modelů) a odchylka bodů v rámci segmentu od jeho náhradního modelu, díky čemuž můžeme stanovit rovinnost případně válcovitost daného segmentu. Výstup programu lze přizpůsobit nadřazenému systému. Návratovými hodnotami mohou být buď přímo změřené hodnoty nebo pouze příznaky, jestli jsou zvolené rozměry v toleranci či nikoliv.

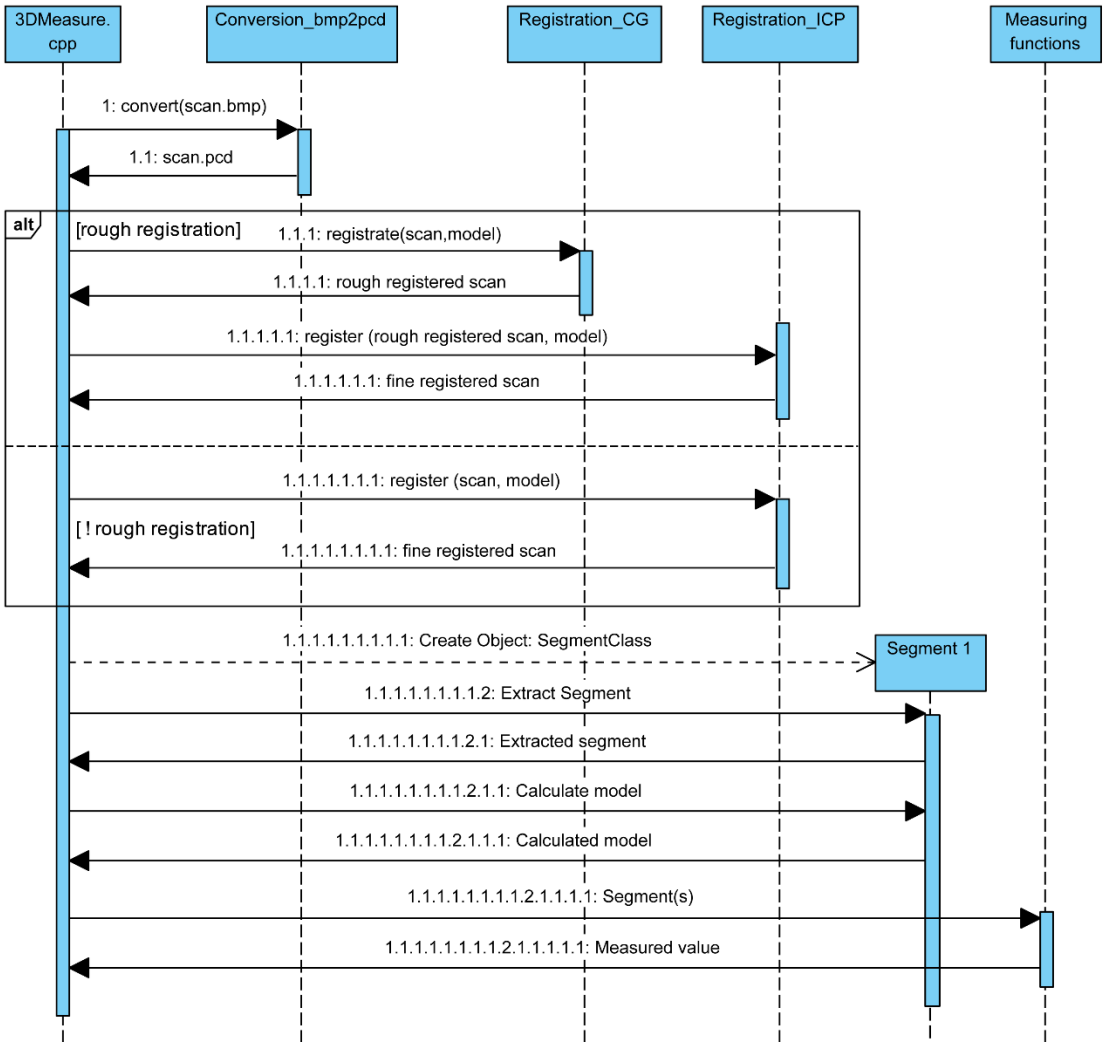
4.2 Struktura programu

Program je členěn do 10 souborů a implementuje třídu SegmentationClass, která slouží k vytváření jednotlivých měřených segmentů. Celý projekt je tvořen následujícími soubory:

- 3DMeasure.cpp
- Conversion_bmp2pcd.cpp
- Read_CSV.cpp
- Registration_CG.cpp
- Registration_ICP.cpp
- Alignment_check.cpp
- SegmentClass.cpp
- Convenient_functions.cpp
- Segment_deviation.cpp
- Segment_distance.cpp
- Segment_angle.cpp
- SegmentClass.h
- 3DMeasure_H.h

Hlavní tělo programu se nachází v souboru 3Dmeasure.cpp a ostatní soubory obsahují jednotlivé dílčí algoritmy, které jsou volány z hlavní funkce. Na začátku každého ze souborů 3DMeasure.cpp, Conversion_bpm2pcd.cpp, Read_CSV.cpp, Registration_CG.cpp, Registration_ICP.cpp a Alignment_check.cpp jsou přehledně seskupeny parametry, pomocí kterých může uživatel ladit proces zpracování dat. Předpokládá se, že program bude potřeba vždy naladit s ohledem na specifický případ použití. Zjednodušený sekvenční diagram programu je zachycen na obrázku 4.1.

V následujících kapitolách popíšu podrobně jednotlivé funkce programu včetně použitých algoritmů z knihovny PCL.



Obrázek 4.1 UML diagram programu

4.2.1 3DMeasure.cpp

Jak již bylo zmíněno výše, tento soubor obsahuje hlavní smyčku programu (main funkci). Tabulka 4.1 obsahuje přehled parametrů a jejich stručný popis, kterými může uživatel ovlivnit proces zpracování dat. Tyto parametry se nacházejí na začátku funkčního souboru 3DMeasure.cpp.

Na začátku programu jsou definovány veškeré potřebné proměnné. Jak je zmíněno na oficiálních webových stránkách PCL knihovny [13], základní proměnná, která slouží pro uchovávání point cloudu, je typu pcl::PointCloud<PointType>. Proměnná PointType určuje formát point cloudu resp. reprezentaci jednotlivých bodů point cloudu. Zjednodušeně řečeno určuje, jaké informace jsou uchovávány ke každému bodu point

cloudu. Například nejzákladnější typ pcl::PointXYZ uchovává pro každý bod pouze tři hodnoty a těmi jsou jeho souřadnice v osách x, y a z. Dalšími implementovanými typy jsou například reprezentace bodu pcl::XYZI, který navíc uchovává i hodnotu intenzity signálu odraženého od skenovaného předmětu nebo pcl::XYZRGB, který kromě souřadnic obsahuje i informace o barvě povrchu v daném místě. Uživatel si může nadefinovat i vlastní typ reprezentace point cloudu. Zvolený typ reprezentace může mít vliv i na pozdější zpracování dat, vzhledem k tomu, že některé procesy mohou pro výpočet vyžadovat specifické rozšiřující informace o každém bodě a nestačí jim pouze jejich souřadnice [x, y, z]. Takovým případem může být detekce významných bodů pomocí Harrisova operátoru ve 3D prostoru, který vyžaduje i informaci o intenzitě odraženého signálu skenerem. Vzhledem k tomu, že point cloudy mohou být velké soubory, je výhodnější pracovat spíše s ukazateli. Ukázku definice některých proměnných typu ukazatel na point cloud, které jsou použity v programu, obsahuje následující úryvek kódu. Tyto proměnné jsou předávány jako parametry funkcím zajišťujícím registraci.

```
pcl::PointCloud<PointType>::Ptr raw_scan(new pcl::PointCloud<PointType>());
pcl::PointCloud<PointType>::Ptr model(new pcl::PointCloud<PointType>());
```

Tabulka 4.1 Parametry 3DMeasure.cpp

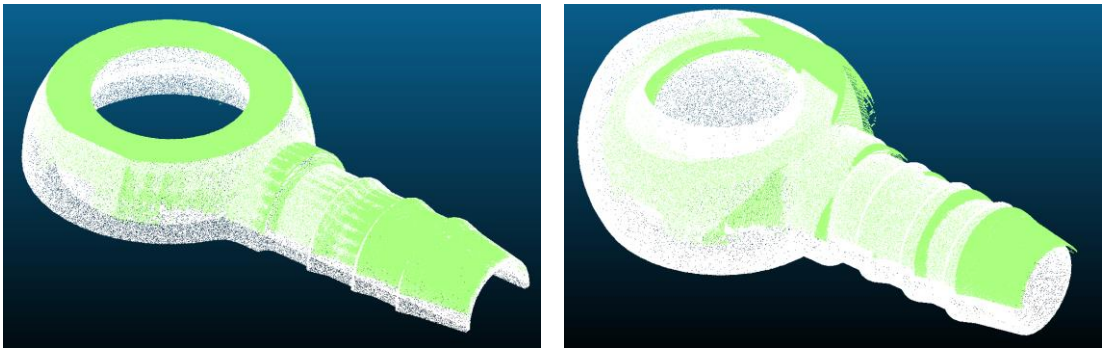
Název parametru	Popis
model_name	Název a cesta k PCD souboru modelu
BMP_name	Název a cesta k BMP souboru skenu
CSV_name	Název a cesta k CSV souboru skenu
CSV_separator	Oddělovač hodnot v souboru CSV
readBMP	True – čtení skenu z BMP souboru False – čtení skenu z CSV souboru
Statistical_outlier_removal	True – Použití funkce Statistical Outlier Removal False – Nepoužívat filtraci falešných bodů
SOR_mean_K	Parametr funkce Statistical Outlier Removal
SOR_Std_dev_mltp	Parametr funkce Statistical Outlier Removal
skip_rough_registration	True – registrace pouze pomocí funkce Registration_ICP False – reg. pomocí funkce Registration CG a Registration_ICP
save_files	True – ukládání průběžných výsledků během procesu False – neukládat žádné průběžné výsledky
Alignment_check	True – provést kontrolu výsledku registrace False – neprovádět kontrolu výsledku registrace
check_with_realignment	True – kontrola výsledku registrace s opravou zarovnání False – kontrola výsledku registrace bez opravy zarovnání

Samotný proces začíná načtením vstupních souborů. Předpokládá se, že soubor obsahující data modelu je už ve formátu PCD, a tudíž obsahuje přímo point cloud.

Z hlediska časové náročnosti je nejvýhodnější, když je model uložen v binárním PCD souboru. Binární PCD soubor je načten mnohonásobně rychleji než ASCII formát PCD souboru. Tento fakt byl ověřen na načítání point cloudu, který byl tvořen 1 milionem bodů. Binární formát byl načten za 47 milisekund, ASCII formát za 3,81 sekund.

Pokud je skenovaný konstrukční díl symetrický, je nutné pracovat pouze s jednou z jeho symetrických částí. Výjimkou je případ, kdybychom výsledný point cloudu skenu skládali z více dílčích skenů, čímž bychom získali sken celého dílu ze všech stran. Pokud bychom použili celý model v kombinaci s částečným skenem, mohly by se vyskytnout problémy při registraci skenu kvůli nejednoznačnosti, ke které ze symetrických částí modelu má být sken zarovnán. Problematiku lépe dokresluje obrázek 4.2.

Surová data získaná ze 3D skeneru mohou být buď ve formátu BMP (Windows Bitmap), nebo CSV. V obou případech, je potřeba vstupní data převést na point cloud, což zajišťují funkce Conversion_bmp2pcd.cpp resp. Read_CSV.cpp



Obrázek 4.2 Bílý – referenční point cloud modelu, zelený – zarovnávaný point cloud skenu.

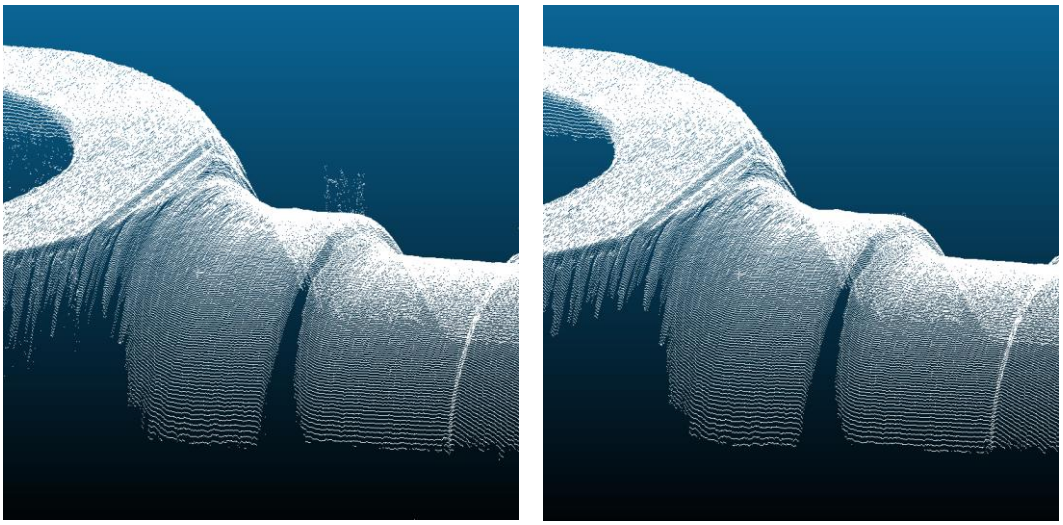
Po načtení skenu konstrukčního dílu můžeme provést filtraci falešných bodů vzniklých při skenování, které by mohly negativně ovlivňovat následný proces zarovnání a měření. Použití funkce však není nezbytné a její využití záleží na uživateli. K filtraci je využita funkce StatisticalOutlierRemoval, která z bodů, jejichž počet nastavuje uživatel pomocí metody setMeanK(), spočítá střední hodnotu vzdálenosti mezi těmito body a její standardní odchylku. Následně jsou odfiltrovány všechny body, které leží mimo interval daný rovnicí

$$\mu \pm \alpha \cdot \sigma \,, \tag{4.1}$$

kde μ je střední vzdálenost, σ je standardní odchylka a α je koeficient nastavovaný uživatelem pomocí metody setStddevMulThresh() [17].

```
pcl::StatisticalOutlierRemoval<PointType> sor;
        sor.setInputCloud(cloudptr);
        sor.setMeanK(50);
        sor.setStddevMulThresh(1.0);
        sor.filter(*aaScan);
```

Příklad úspěšného výsledku funkce StatisticalOutlierRemoval můžeme vidět na obrázku 4.3, kdy došlo k odstranění téměř všech falešných bodů, které vznikly kvůli odrazům na hraně skenovaného předmětu.



Obrázek 4.3 Naskenovaný díl před (nalevo) a po (napravo) použití funkce StatisticalOutlierRemoval

Dále jsou z funkce 3DMeasure.cpp zavolány funkce zajišťující zarovnání skenu vůči modelu. Použití pomocného hrubého zarovnání záleží na konkrétním případu využití tohoto programu. V některých případech není potřeba a stačí použít pouze funkci Registration_ICP. Může se však stát, že bez předběžného hrubého zarovnání nezvládne funkce Registration_ICP dojít k úspěšnému výsledku, a proto je provedení hrubého zarovnání nezbytné. Hrubé zarovnání zajišťuje funkce Registration_CG a to, zda je funkce použita záleží na nastavení příslušného parametru uživatelem. Výsledek zarovnání lze zkontrolovat a pomocí funkce Alignment_check. Cílem této funkce je ošetření případu, kdy iterativní algoritmus použitý ve funkci Registration_ICP konverguje pouze k lokálnímu minimu a tím pádem nedojde k úspěšnému zarovnání. Pokud funkce vyhodnotí zarovnání jako chybné, tak dojde k ukončení celého programu.

Po úspěšném zarovnání dochází k vytváření jednotlivých objektů typu SegmentationClass. Parametry a procesy týkající se jednotlivých objektů jsou nastavovány a spouštěny pomocí metod této třídy. Následující kód názorně ukazuje práci s objekty typu SegmentationClass. Význam jednotlivých metod je vysvětlen v kapitole 4.2.6.

Jakmile jsou vytvořeny všechny požadované instance třídy SegmentationClass, můžeme přikročit k funkcím zajišťujícím samotné měření rozměrů naskenovaného dílu. Těmito funkcemi jsou Segment_deviation, Segment_distance a Segment_angle. Funkce segment_deviation.cpp měří odchylku jednotlivých bodů daného segmentu od nalezeného náhradního modelu, Segment_distance měří vzdálenost mezi dvěma segmenty (předpokládají se dva segmenty obsahující rovinu) a Segmentation_angle.cpp

počítá úhlovou odchylku dvou segmentů. Tyto funkce jsou podrobněji popsány v následujících kapitolách.

Změřené hodnoty jsou následně upraveny do požadovaného formátu a předány do nadřazeného systému.

```
SegmentClass segment1(registered_scan);
      segment1.setLimits({26.5, 39.0, 139.0, 198.0, -2.3, -1.28});
      segment1.extractSegment();
      if (!segment1.ExtractedSegment->empty()) {
            segment1.setMethod(pcl::SAC_RANSAC);
            segment1.setModel(pcl::SACMODEL_CYLINDER);
            segment1.setOptimizeCoeffs(true);
            segment1.setIterations(100);
            segment1.setThreshold(0.04);
            segment1.setRadiusLimits({ 4.5, 6.0 });
            segment1.setAngleRest(false);
            segment1.setModelRestAxis(axis);
            segment1.setEpsilonAngle(0.1);
            segment1.calculateModel();
      }
```

4.2.2 Conversion_bmp2pcd.cpp

Vzhledem k tomu, že testovací sada dat získaná pomocí skeneru Keyence LJ-X8080 připojenému ke kontroléru LJ-X8000 byla ve formátu BMP, bylo potřeba nejprve provést konverzi obrazových dat na prostorová data point cloudu. Konverzi dat zajišťuje právě tato funkce. Pro lepší pochopení toho, co tato funkce dělá, je potřeba začít popisem formátu souborů BMP.

Za vznikem BMP formátu stojí firma Microsoft, která jej vyvinula pro ukládání grafických souborů. Soubory mají přesně danou strukturu obsažených dat, některé části však nejsou povinné pro všechny varianty. Jednotlivé varianty se odvíjí mimo jiné od počtu bitů, které reprezentují jeden pixel obrazu. BMP podporuje obrázky, jejichž pixely jsou reprezentovány 1, 2, 4, 8, 16, 24 nebo až 32 bity [18]. Soubor generovaný skenerem LJ-V7080 generuje 8bitový RGB formát, proto se ostatním variantám nebudu více věnovat.

BMP soubor obsahuje hlavičku souboru (14 bytů) a informační hlavičku (40 bytů) za kterými následují samotná obrazová data. Z hlediska tohoto projektu nejdůležitějšími parametry obsaženými v obrazové hlavičce jsou šířka a výška uloženého obrazu. Hodnota šířky a výšky obrazu je typu int (4 byty), přičemž hodnota šířky se nachází na 18. bytu od začátku souboru a výška na 22. bytu od začátku soboru. Vynásobením těchto hodnot je získáno celkové rozlišení obrazu a tím pádem i celková velikost obrazových dat uložených za hlavičkou BMP souboru. Následující úryvek kódu zajišťuje načtení obrazových dat pro další zpracování.

```
FILE* f = fopen(filename, "rb");       //scan.bmp
        if (f == NULL)
            throw "Argument ERROR";

// read the 54-byte file header (14) + information header (40)
unsigned char info[54];
fread(info, sizeof(unsigned char), 54, f);

// extract image height and width from header
int width = *(int*)&info[18];
int height = *(int*)&info[22];

int size = 3 * width * height;
unsigned char* data = new unsigned char[size];

// extract image data
fread(data, sizeof(unsigned char), size, f);
fclose(f);
```

Samotná obrazová data jsou organizována za sebou po jednotlivých řádcích, přičemž se začíná od nejspodnějšího řádku. Proto vzhledem k postupnému čtení hodnot jednotlivých pixelů může dojít k zrcadlovému obrácení point cloudu oproti tomu, jak je zobrazen v BMP souboru. Proto lze pomocí parametru BMP_flip zvolit, zda mají být data načtená z BMP souboru zrcadlově převráceny.

Řádek a sloupec, ve kterém se pixel nachází, určuje hodnotu x a y souřadnice daného bodu point cloudu. Souřadnice v ose z je ukryta v hodnotě zeleného kanálu daného pixelu. Díky tomu můžeme provést první filtrování dat pomocí stanovení limitu vzdálenosti, která data mají být považována ještě za platná ve vztahu ke konstrukčnímu dílu a která mohou představovat již podložku na které díl během skenování leží. To může zmenšit objem dat, převáděných na point cloud a tím pádem urychlit běh programu. Hodnota parametru BMP_Filter_limit musí být zjištěna empiricky. Povšimněme si, že barevné kanály RGB každého pixelu jsou uloženy v pořadí BGR.

```
for (int i = 0; i < 3 * width * height; i += 3) {//filtrace podlozky
        B = (int)data[i];
        G = (int)data[i + 1];
        R = (int)data[i + 2];
        if ((B == 0 && G == 0 && R == 0) || (G <= BMP_Filter_limit)) {
            data[i] = (unsigned char)255;
            data[i + 1] = (unsigned char)255;
            data[i + 2] = (unsigned char)255;
}
```

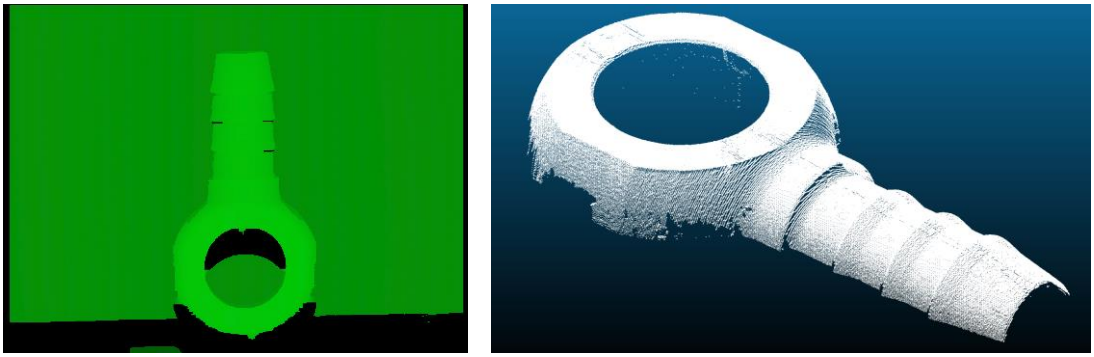
Závěrečným krokem funkce Conversion_bmp2pcd.cpp je samotné převedení obrazových hodnot na prostorová data point cloudu. To zajišťuje následující kód. Způsob převodu hodnot jednotlivých barevných kanálů pixelu na souřadnici odpovídajícího bodu v ose z je dán takto výrobcem skeneru. Hodnoty proměnných height_resolution, height_lower_value, x_resolution a y_resolution záleží na nastavení skeneru a ostatních perfierií.

```
pcl::PointCloud<PointType> cloud;
for (int i = 0; i < height; ++i)
{
   for (int j = 0; j < width * 3; j += 3)
     {
       B = (int)data[(i * width * 3) + j];
       G = (int)data[(i * width * 3) + j + 1];
       R = (int)data[(i * width * 3) + j + 2];

       if ((B == 255 && G == 255 && R == 255)) {
           continue;
       }

       heigth_data = (G << 7) | ((R & 0x07) << 4) | (B & 0x0f);
       coord_z = (height_resolution * heigth_data) + height_lower_value;
       cloud.points[cnt].x = (j / 3) * x_resolution;
       cloud.points[cnt].y = i * y_resolution;
       cloud.points[cnt].z = coord_z;
            ++cnt;
     }
}
```

Proměnná cloud tedy obsahuje point cloud naskenovaného konstrukčního dílu.



Obrázek 4.4 Nalevo původní BMP soubor získaný ze skeneru, napravo point cloud získaný konverzí dat z formátu BMP

4.2.3 Read_CSV.cpp

Soubor získaný ze skeneru však nemusí být vždy ve formátu BMP. Běžným formátem pro ukládání hodnot je CSV, kde jsou jednotlivé hodnoty odděleny čárkami, případně jiným oddělovačem (např. středníkem, tabulátorem…). Tato funkce zajišťuje čtení dat ze souboru formátu CSV a uložení dat do odpovídajícího point cloudu. Parametry předávané této funkci jsou cesta k souboru CSV, ukazatel na point cloud, kam mají být hodnoty uloženy a předpokládaný oddělovač hodnot souboru v CSV. Funkce je naprogramována pro zpracování dat point cloudů, jejichž bodová reprezentace odpovídá typu pcl::PointXYZ. Uživatel může nastavit parametry uvedené v tabulce 4.2.

Tabulka 4.2 Read_CSV.cpp

Název parametru	Popis
x_coeff	Koeficient k výpočtu souřadnice bodu v ose x. Hodnota je dána šířkou profilu a počtem bodů v rámci profilu
y_coeff	Koeficient k výpočtu souřadnice bodu v ose y. Hodnota je dána rychlostí pohybu v ose y a snímací frekvencí.
background_limit	Slouží k filtraci pozadí od skenovaného dílu na základě hodnoty v ose Z, která je zapsána v CSV souboru.

4.2.4 Registration_CG.cpp

Jakmile je k dispozici model i sken konstrukčního dílu ve formě point cloudu, lze přistoupit k zarovnání skenu vůči modelu. Jak již bylo zmíněno výše, významem funkce Registration_CG je provedení hrubého zarovnání, které může pomoct k dosažení lepších výsledků registrační funkce Registration_ICP. V této kapitole bude věnována pozornost algoritmům z knihovny PCL, které jsou využity pro proces hrubého zarovnání. Schéma funkce Registration_CG znázorňuje diagram 4.5. Na začátku souboru Registration_CG.cpp jsou definovány parametry, pomocí kterých může uživatel ovlivnit registrační proces. Tyto parametry a jejich stručný popis je uveden v tabulce 4.3.

Tabulka 4.3 Parametry Registration_CG.cpp

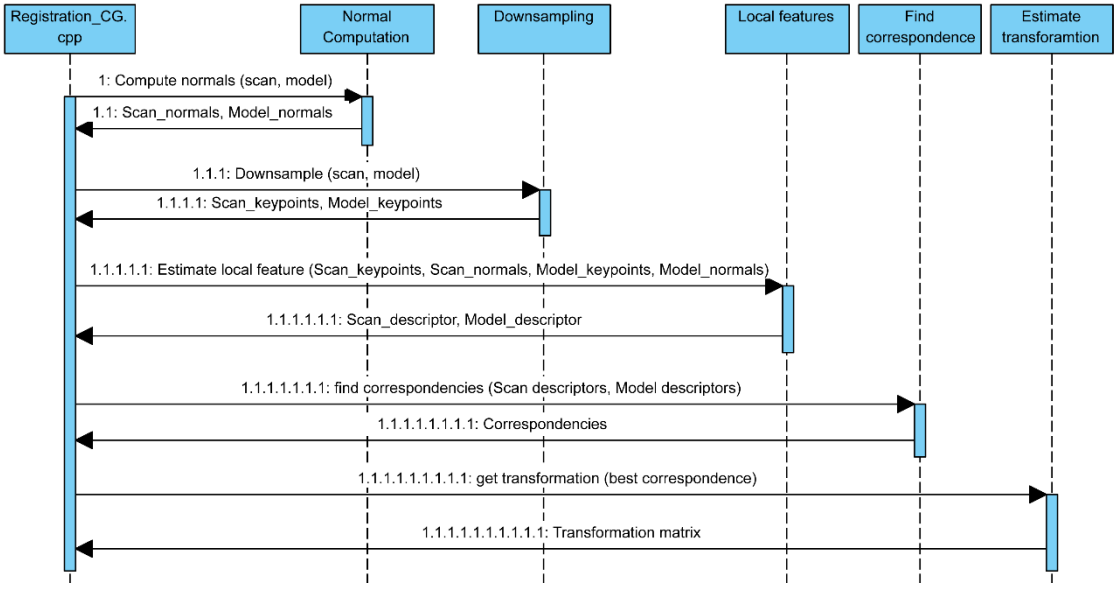
Název parametru	Popis
CG_use_hough	True – použití funkce Houhg 3D pro třídění korespondencí False – použití funkce GeometricConsistency
CG_model_ss	Krok vzorkovacího filtru pcl::UniformSampling pro model
CG_scan_ss	Krok vzorkovacího filtru pcl::UniformSampling pro sken
CG_rf_rad	Poloměr oblasti pro výběr bodů pro výpočet LRF
CG_descr_rad	Poloměr oblasti pro výběr bodů pro výpočet deskriptoru
CG_cg_size	Velikost buňky uvnitř Houghovy funkce
CG_cg_thresh	Minimální počet korespondencí dané buňky uvnitř Houghovi funkce, aby byla považována za platnou
CG_norm_K_search_neighbour	Počet sousedních bodů použitých pro výpočet normál
CG_squared_descriptor_distance	Maximální vzdálenost dvou deskriptorů, aby mohly tvořit pár

Funkce Registration_CG.cpp vychází ze vzorového příkladu použití knihovny PCL, které jsou volně dostupné na oficiálních stránkách knihovny [13]. Výsledný kód je však upraven potřebám této práce. Funkce pracuje na principu hledání korespondencí mezi modelem a skenem. Původní myšlenkou je hledání všech instancí modelu ve scéně, avšak v mém případě tvoří celou scénu pouze jeden naskenovaný díl. Výsledkem funkce je nalezení nejlepší korespondence a na jejím základě určení transformační matice, která transformuje point cloud skenu do pozice blížící se pozici modelu. Výsledná transformační matice má rozměr 4x4, přičemž první tři řádky a tři sloupce odpovídají

matici otočení a první tři řádky čtvrtého sloupce odpovídají translaci. Poslední řádek zůstává vždy stejný a neměnný. Například matice

$$T = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 5 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.2}$$

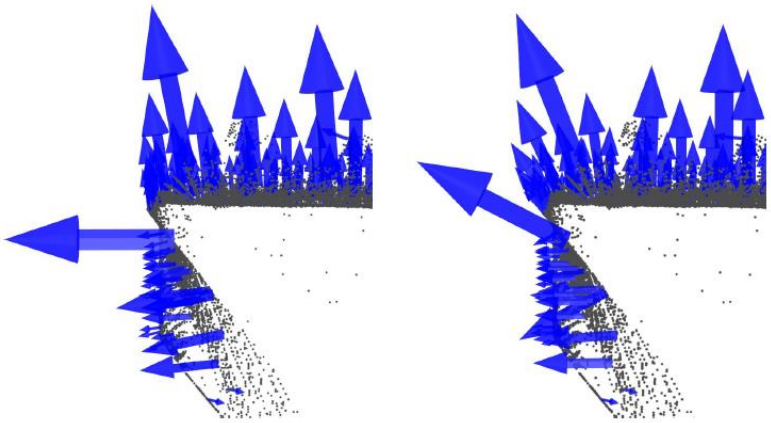
zajistí otočení kolem osy z o úhel θ proti směru hodinových ručiček a posunutí ve směru osy y o hodnotu 5 a ve směru osy z o hodnotu 4.



Obrázek 4.5 Digram Registration_CG.cpp

Algoritmus hledá korespondence mezi modelem a skenem na základě porovnávání lokálních vlastností každého z nich. Pro stanovení lokálních vlastností je využit deskriptor SHOT (Signature of Histograms of OrienTations), který provádí výpočet na základě bodů point cloudu a jim příslušných normál, které reprezentují orientaci plochy v těchto bodech. Proto musíme nejprve nechat spočítat normály pro model i pro sken.

Pro stanovení normál musí uživatel zvolit, z jak velkého okolí každého bodu p_q bude výsledná normála vypočtena. Tento parametr můžeme označit jako *k*-okolí bodu p_q a lze zvolit jako počet *k* nejbližších sousedních bodů kolem p_q, nebo jako poloměr koule *r* se středem v bodě p_q, která ohraničuje všechny sousední body, které mají být brány při výpočtu normály v potaz. Jak zvolit vhodné *k* nebo *r* však nelze jednoduše určit. Pokud zvolíme příliš velkou hodnotu, může docházet k potlačení vlivu menších lokálních zakřivení nebo ke zkreslení na hranách objektu. Situaci lépe znázorňuje obrázek 4.6, na kterém vidíme porovnání mezi případem, kdy je hodnota *k* nebo *r* zvolena správně a případem, kdy je hodnota příliš vysoká [19].



Obrázek 4.6 Nalevo správně zvolená hodnota *k* nebo *r*, napravo příliš velká hodnota *k* nebo *r* [19]

Následující postup výpočtu normál popisuje autor knihovny PCL ve své disertační práci [19]. Body tvořící okolí bodu p_q můžeme označit jako množinu P^k. Normála reprezentující orientaci povrchu je stanovena na principu hledání roviny tečné k povrchu v bodě p_q, která je reprezentována bodem *x*, normálovým vektorem $\vec{n}$ a vzdálenost bodu $p_i \in P^k$ je stanovena jako

$$d_i = (p_i - x) \cdot \vec{n}. \tag{4.3}$$

Předpis hledané roviny tak odpovídá obecnému tvaru rovnice roviny

$$ax + by + cz + d = 0, \tag{4.4}$$

kde *x*, *y* a *z* jsou odpovídající souřadnice bodu *x* a *a*, *b*, *c* jsou jednotlivé složky normálového vektoru $\vec{n}$. Určení bodu *x* a vektoru $\vec{n}$ je stanoveno pomocí metody nejmenších čtverců, tak aby byla splněna podmínka $d_i = 0$. Bod *x* odpovídá těžišti množině bodů P^k, které spočítáme podle rovnice

$$\mathrm{x} = \bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i. \tag{4.5}$$

Vektor $\vec{n}$ získáme výpočtem vlastních čísel a vlastních vektorů kovarianční matice C množiny bodů P^k, kterou získáme podle

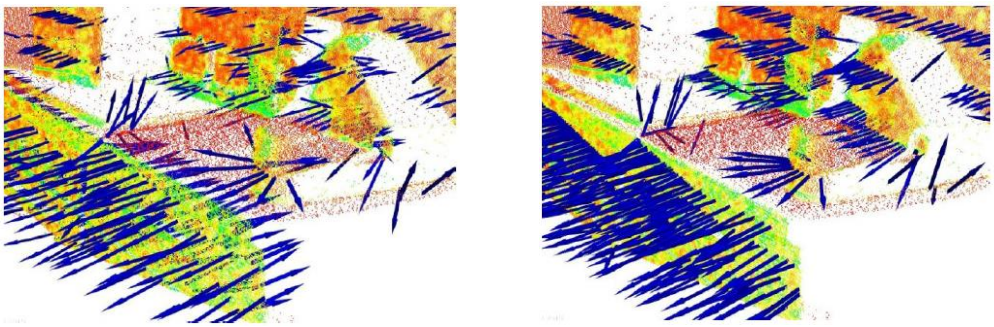
$$\mathrm{C} = \frac{1}{k} \sum_{i=1}^{k} (p_i \cdot \bar{p}) \cdot (p_i \cdot \bar{p})^T, \qquad C \cdot \overrightarrow{v_j} = \lambda_j \cdot \overrightarrow{v_j}, j \in \{0,1,2\}. \tag{4.6}$$

Hledaný normálový vektor roviny $+\vec{n}$ případně $-\vec{n}$ odpovídá vlastnímu vektoru $\overrightarrow{v_0}$, který odpovídá vlastnímu číslu λ_0 při splnění podmínky $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$. Neexistuje však matematické řešení, jak určit, jestli výsledný vektor odpovídá normálovému vektoru $+\vec{n}$ nebo $-\vec{n}$. Avšak vzhledem k tomu, že z podstaty principu skenování víme, ze které

strany se na plochu díváme, stačí vypočtené normály orientovat tak, aby odpovídaly podmínce

$$\overrightarrow{n_i} \cdot (v_p - p_i) > 0, \tag{4.7}$$

kde v_p označuje bod, ze kterého byl sken pořízen (umístění skeneru v prostoru). Problematiku orientace normál znázorňuje obrázek 4.7.



Obrázek 4.7 Nalevo vypočtené normály před úpravou orientace, napravo po úpravě orientace [19]

Vypočtená vlastní čísla kovarianční matice jsou rovněž použity pro výpočet zakřivení povrchu v bodě p. Zakřivení σ_p je stanoveno podle rovnice

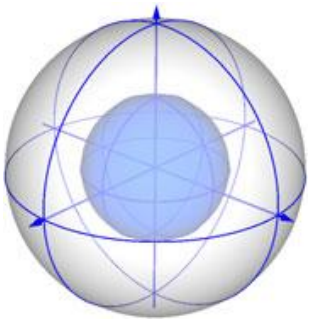
$$\sigma_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \tag{4.8}$$

přičemž nízká hodnota znamená, že body množiny P^k leží v blízkosti případně přímo odpovídají rovině tečné k povrchu v daném bodě p_q [19].

Vzhledem k výpočetní náročnosti hledání lokálních vlastností není vhodné provádět výpočet pro všechny body point cloudu. Proto jsou v dalším kroku funkce Registration_CG.cpp nalezeny klíčové body a lokální vlastnosti jsou stanoveny pouze v těchto bodech. Z hlediska efektivity je nejvýhodnější nalézt co nejmenší množství bodů, pro které lokální vlastnosti počítáme, které však stále dostatečně přesně popisují původní point cloud. V tomto konkrétním případě je pro získání významných bodů použit objekt pcl::UniformSampling<PointType>, který najde významné body za využití 3D voxelové struktury, přičemž rozlišení struktury (velikost jednotlivých voxelů) je nastavitelný parametr, který zadává uživatel. Velikost tohoto parametru je potřeba stanovit empiricky podle konkrétního příkladu použití. Čím větší hodnota bude, tím méně významných bodů získáme. PCL nabízí i jiné algoritmy pro výpočet významných bodů jako je například Harris 3D, nebo SIFT 3D. Jak již bylo zmíněno výše, tyto algoritmy pracují s point cloudy, jejichž body nesou informaci i o intenzitě odraženého signálu. Vzhledem k tomu, že tento údaj u point cloudu modelu není dostupný, není použití těchto dvou alternativ vhodné.

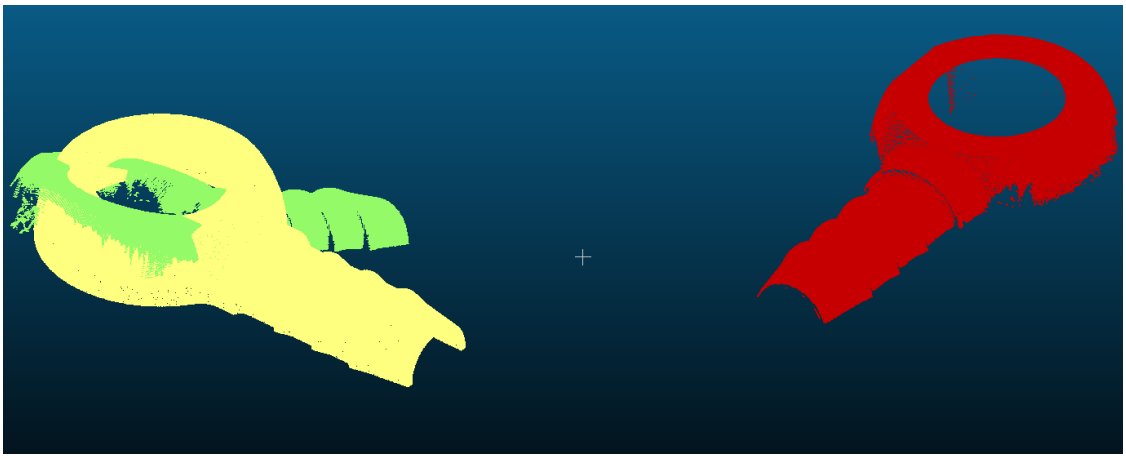
Výpočet lokálních vlastností je realizován pomocí deskriptoru SHOT, který kombinuje výhody deskriptorů založených na vyhodnocování charakteristických rysů oblasti, což přináší vysokou rozlišovací schopnost, ale také citlivost na šum a jiné nežádoucí efekty, nebo těch, které popisují danou oblast pomocí histogramu zvolených parametrů, což přináší robustnost, ale nižší rozlišovací schopnost [20]. Pro každý významný bod je nejprve stanoven lokální souřadnicový systém (v zahraniční literatuře označován jako Local Reference Frame, proto bude dále použita zkratka LRF) spočítaný z jeho okolních bodů. Velikost uvažovaného okolí stanovuje uživatel pomocí poloměru koule, která má střed v tomto bodě. LRF by měl být unikátní a jednoznačný, aby bylo možné dosahovat stejných výsledků při změně otočení nebo měřítka. Spousta deskriptorů však používá lokální souřadnicové systémy, které jsou buď unikátní, nebo jednoznačné, a proto musí kombinovat více LRF najedou, což může zvyšovat výpočetní náročnost [20]. Proto autoři deskriptoru SHOT navrhli vlastní způsob výpočtu LRF, který zajišťuje unikátnost a jednoznačnost orientace. Poté je kolem významného bodu vytvořen kulovitý izotropní prostor, který je rozdělen na 32 stejně velkých oblastí (viz obrázek 4.8). Pro každou oblast je spočítán 1dimenzionální lokální histogram, jehož hodnotami jsou cosiny úhlů svíraných mezi normálami bodů z daného výřezu koule a úhlu normály odpovídající významnému bodu, pro nějž hledáme lokální vlastnosti [20].

Po výpočtu lokálních vlastností modelu i skenu následuje hledání jednotlivých korespondencí mezi těmito point cloudy. Hledání korespondencí probíhá na základě měření vzdálenosti vypočtených deskriptorů. Pokud je vzdálenost deskriptorů menší než uživatelem nastavený práh (vzdálenost deskriptoru SHOT se pohybuje v intervalu od 0 do 1 [13]), je korespondence považována za platnou. Jednotlivé instance modelu ve scéně a jim odpovídající transformační matice jsou nalezeny pomocí objektu pcl::Hough3DGrouping, který roztřídí platné korespondence mezi modelem a skenem získané v předchozím kroku. Vzhledem k tomu, že scéna v tomto případě obsahuje pouze jednu instanci modelu, považujeme instanci, které odpovídá nejvíc korespondencí, za výsledek a její transformační matici aplikujeme na point cloud skenu. Nastavitelnými parametry objektu typu pcl::Hough3DGrouping je HouhgBinSize a HoughThreshhold. Prvně zmíněný parametr určuje velikost jednotlivých buněk uvnitř Houghovy funkce, do kterých jsou korespondence rozdělovány. Čím nižší tato hodnota bude, tím přesnější je rozdělení korespondencí mezi jednotlivé instance. HoughThreshhold pak specifikuje minimální nutný počet korespondencí odpovídajících dané buňce, aby mohla být považována za relevantní. Tento parametr musí mít hodnotu 3 nebo vyšší, což vychází ze skutečnosti, že pro určení přesně polohy a orientace libovolného objektu ve 3D prostoru jsou potřeba nejméně 3 body odpovídající tomuto objektu.



Obrázek 4.8 Rozdělení prostoru deskriptoru SHOT kolem významného bodu. Pro přehlednost jsou znázorněny pouze 4 místo 8 rozdělení podél azimutu. [20]

Ukázku výsledku funkce Registration_ICP.cpp ukazuje obrázek 4.9. Vidíme, že zarovnání je opravdu pouze přibližné a je potřeba provést ještě přesné dorovnání pomocí funkce Registration_ICP.cpp.



Obrázek 4.9 Žlutý – model, červený – původní sken, zelený – hrubě zarovnaný sken

4.2.5 Registration_ICP.cpp

Dalším krokem procesu je přesné dorovnání point cloudu skenu vůči modelu pomocí implementovaného algoritmu ICP. Algoritmus může být použit rovnou i na původní point cloud skenu, ale obecně vzato je lepší pracovat s již hrubě zarovnaným point cloudem. Nevýhodu této metody totiž může být, že nalezený nejlepší výsledek (co nejmenší odchylka mezi skenem a modelem) odpovídá pouze lokálnímu minimu [19]. Na začátku souboru Registration_ICP.cpp jsou definovány parametry, pomocí kterých uživatel může nastavit registrační algoritmus ICP. Tyto parametry jsou uvedeny v následující tabulce 4.4.

Tabulka 4.4 Parametry Registration_ICP.cpp

Název parametru	Popis
ICP_downsample	True – snížit počet bodů point cloudu skenu a modelu False – nesnižovat počet bodů skenu a modelu
ICP_downsample_leaf_size	Krok vzorkovacího filtru
ICP_norm_K_search_neighbour	Počet sousedních bodů použitých pro výpočet normál
ICP_TransformationEpsilon	Práh odchylek dvou po sobě jdoucích iterací algoritmu ICP
ICP_MaxCorrespondenceDistance	Maximální vzdálenost korespondenčního páru sken – model
ICP_MaximumIterations	Maximální počet iterací algoritmu ICP

Tato funkce také vychází ze vzorového příkladu, který je volně dostupný na oficiálních stránkách knihovny PCL, opět však byla upravena potřebám této práce. Na začátku mohou být point cloudy skenu a modelu vyfiltrovány pomocí objektu pcl::VoxelGrid<PointType>, který sníží počet bodů zarovnávaných point cloudů. Jedná se však pouze o krok zajišťující urychlení procesu, protože je výhodnější zarovnávat nižší počet bodů a výslednou nalezenou transformaci na závěr aplikovat na původní detailní point cloud skenu. Dalším krokem je výpočet normál pro všechny body obou vyfiltrovaných point cloudů. Způsob výpočtu normál byl již detailně popsán v kapitole 4.2.3.

Samotný proces zarovnání a hledání výsledné transformace zajišťuje objekt typu pcl::IterativeClosestPointNonLinear<PointNormalT, PointNormalT>. Uživatel může nastavit maximální vzdálenost korespondencí mezi modelem a skenem, které mají být brány v potaz, maximální počet iterací a hodnotu ε, která definuje odchylku kvadratické vzdálenosti transformací dvou po sobě jdoucích iterací zarovnávacího algoritmu. Pokud je tato vzdálenost menší než práh ε, dojde k ukončení algoritmu a výsledek je považován za úspěšný i ve chvíli, kdy ještě nebylo dosaženo maximálního počtu iterací. Po ukončení algoritmu je nalezená transformace aplikována na point cloud skenu. Využití implementovaného algoritmu ICP ukazuje následující ukázka kódu.

```
pcl::IterativeClosestPointNonLinear<PointNormalT, PointNormalT> reg;
reg.setTransformationEpsilon(ICP_TransformationEpsilon);
reg.setMaxCorrespondenceDistance(ICP_MaxCorrespondenceDistance);
reg.setMaximumIterations(ICP_MaximumIterations);
reg.setInputSource(points_with_normals_scan);  //Scan
reg.setInputTarget(points_with_normals_model); //Model
reg.align(*reg_output);                        //Estimate alignment
targetToSource = reg.getFinalTransformation(); //Final transformation
pcl::transformPointCloud(*cloud_scan, *output, targetToSource);
```

4.2.6 Alignment_check.cpp

Jak již bylo zmíněno v kapitole 4.2.1, tato funkce zajišťuje kontrolu výsledku procesu

registrace. Proces kontroly může uživatel ovlivnit pomocí následujících parametrů uvedených v tabulce 4.5.

Tabulka 4.5 Parametry Alignment_check.cpp

Název parametru	Popis
AC_centre_tolerance	Tolerance vzdálenosti geometrického středu skenu a modelu
AC_major_axis_tol	Tolerance úhlu mezi hlavní osou skenu a modelu
AC_middle_axis_tol	Tolerance úhlu mezi první vedlejší osou skenu a modelu
AC_minor_axis_tol	Tolerance úhlu mezi druhou vedlejší osou skenu a modelu
AC_iterations	Počet iterací opravného zarovnání
AC_theta	Krok otočení kolem osy z

Funkce je implementována ve dvou variantách jejichž předpisy se nachází v následujícím úryvku kódu. Argumenty první varianty jsou ukazatel na point cloud modelu a zarovnaného skenu. Druhá se od té první liší tak, že v případě zjištění chybného zarovnání se pokusí proces zarovnání iterativně zopakovat. Opravný algoritmus je ukončen, když dojde ke správnému zarovnání, nebo je dosaženo maximálního počtu iterací daných parametrem AC_Iterations. Druhá varianta funkce proto vyžaduje navíc ještě ukazatel na point cloud, do kterého se uloží opravený správně zarovnaný point cloud skenu. Pokud funkce vyhodnotí zarovnání jako chybné, resp. jej nezvládne ani opravit, je vrácena hodnota -1.

```
int AlignmentCheck(const pcl::PointCloud<PointType>::Ptr cloud_model,
           const pcl::PointCloud<PointType>::Ptr cloud_scan_to_check);

int AlignmentCheck(const pcl::PointCloud<PointType>::Ptr cloud_model,
           const pcl::PointCloud<PointType>::Ptr cloud_scan_to_check,
           pcl::PointCloud<PointType>::Ptr output);
```

Kontrola úspěšného zarovnání vychází z porovnání geometrického středu zarovnávaného skenu a modelu a kontroly úhlů mezi jejich hlavními a dvěma vedlejšími osami. Geometrický střed je bod, jehož souřadnice získáme jako průměr krajních bodů skenu a modelu v osách x, y a z. Osami skenu a modelu se rozumí osy kvádru opsaného těmto point cloudům. K výpočtu těchto os je využita následující funkce knihovny PCL.

```
pcl::MomentOfInertiaEstimation <pcl::PointXYZ> feature_extractor;
feature_extractor.setInputCloud(cloud_model);
feature_extractor.compute();
feature_extractor.getEigenVectors(major_vector_m, middle_vector_m,
                                  minor_vector_m);
```

Nutno připustit, že tento způsob kontroly není ze své podstaty bezchybný, protože geometrické středy skenu i modelu sice mohou ležet na stejných souřadnicích a jejich

jednotlivé osy mohou být shodně orientované, není však zaručeno, že sken není otočený o 180° kolem jedné z vypočtených os. Během vývoje a testování programu však k tomuto případu nedošlo a lze ho proto označit za nepravděpodobný. K řešení tohoto problému se nabízí využití těžiště jednotlivých point cloudů, avšak tento parametr může být příliš ovlivněn nedokonalostmi naskenovaného point cloudu (např. vliv mrtvých úhlů skeneru) a rozdílností hustoty point cloudu modelu a skenu. Detekce geometrického středu z krajních bodů point cloudu je však naopak náchylná na přítomnost falešných bodů, které by se nacházely na okraji point cloudu. Proto není doporučeno používat funkci Alignment_check bez předchozího použití funkce StatisticalOutlierRemoval (viz kapitola 4.2.1).

Opravný mechanismus zarovnání, který je implementován ve druhé variantě funkce Alignment_check iterativně transformuje původní chybně zarovnaný sken a po každé transformaci je provedeno nové zarovnání pomocí funkce Registration_ICP. Poté je výsledek registrační funkce vyhodnocen pomocí porovnání geometrických středů a v případě úspěšného zarovnání dojde k ukončení algoritmu a opravený výsledek je uložen do odpovídající proměnné, se kterou je dále pracováno v hlavní funkci celého programu. Tato metoda opravy využívající princip hrubé síly není nejefektivnějším možným řešením, avšak na základě testování ji lze považovat alespoň za dostatečně robustní řešení.

4.2.7 SegmentClass.cpp

Pro práci se segmenty point cloudu skenu byla vytvořena třída SegmentClass. Tato třída implementuje metody sloužící k extrakci segmentu z point cloudu a k výpočtu jeho náhradního modelu. Pro výpočet náhradního modelu využívá třída SegmentClass funkce knihovny PCL. Kompletní přehled metod a proměnných této třídy znázorňuje diagram v příloze této práce. V této kapitole budou popsány pouze stěžejní části této třídy.

Na začátku je potřeba objektu předat zdrojový point cloud, ze kterého má být segment vyříznut a limity oblasti, ve které se požadovaný segment nachází. To zajišťují buď samotné konstruktory třídy, nebo lze použít příslušné metody. Samotnou extrakci zajišťuje metoda SegmentClass::extractSegment(), která vybere všechny body ze zdrojového point cloudu, které se nachází uvnitř dané oblasti, která je specifikována minimální a maximální hodnotou v osách x, y a z.

Po extrakci segmentu ze zdrojového point cloudu lze přejít k výpočtu jeho náhradního modelu. Třída v tuto chvíli umožňuje pracovat pouze s rovinnými a válcovitými segmenty, ačkoliv knihovna PCL podporuje i další jednouché tvary jako je například přímka, koule, nebo kužel. K výpočtu náhradního modelu jsou využity objekty z knihovny PCL. Pro výpočet modelu roviny je použit objekt pcl::segObject a pro výpočet válce se používá objekt pcl::segObjectNormals, který při výpočtu využívá i normály. Toto rozdělení vychází z faktu, že každý objekt umí pracovat jen s některými modely definovanými v PCL. K hledání náhradního modelu je pak využit algoritmus RANSAC,

jehož výhodou je rychlost a robustnost [21]. V PCL knihovně jsou však implementovány další algoritmy jako jsou například PROSAC, RRANSAC, RMSAC a další.

```
pcl::SACSegmentation<pcl::PointXYZ> segObject;
pcl::SACSegmentationFromNormals<pcl::PointXYZ,pcl::Normal>
     segObjectNormals;
```

Princip metody RANSAC bude popsán na příkladu hledání náhradního modelu roviny [19]. V prvním kroku jsou náhodně vybrány tři body, které však neleží ve stejné přímce. Volba tří bodů vychází z toho, že tři body již definují rovinu. Kdybychom však hledali například model přímky, stačily by pouze dva body. Z těchto tří bodů spočítáme koeficienty obecné rovnice roviny (4.4). V dalším kroku spočítáme pro každý bod segmentu jeho vzdálenost od této roviny. Na základě podmínky

$$0 \leq l \leq t, \tag{4.9}$$

kde l je vzdálenost bodu od roviny a t je tolerance nastavená uživatelem, rozdělíme body na takzvané *inliers* a *outliers*. Jako *inliers* označujeme body, které odpovídají aktuálnímu modelu roviny, a tudíž splňují podmínku (4.9), *outliers* jsou body, které tuto podmínku nesplňují. Na závěr je potřeba vypočítat skóre, podle kterého je hodnocena kvalita aktuálního modelu roviny. Parametrem vhodným pro stanovení takového skóre může být právě počet *inliers*. Celý proces se zopakuje k-krát, přičemž hodnota k záleží na uživateli. Po provedení všech iterací je vybrán model s nejvyšším skóre a z jeho *inliers* jsou za využití metody nejmenších čtverců vypočteny koeficienty výsledného náhradního modelu roviny [19].Vhodnou hodnotu počtu iterací k můžeme určit pomocí rovnice

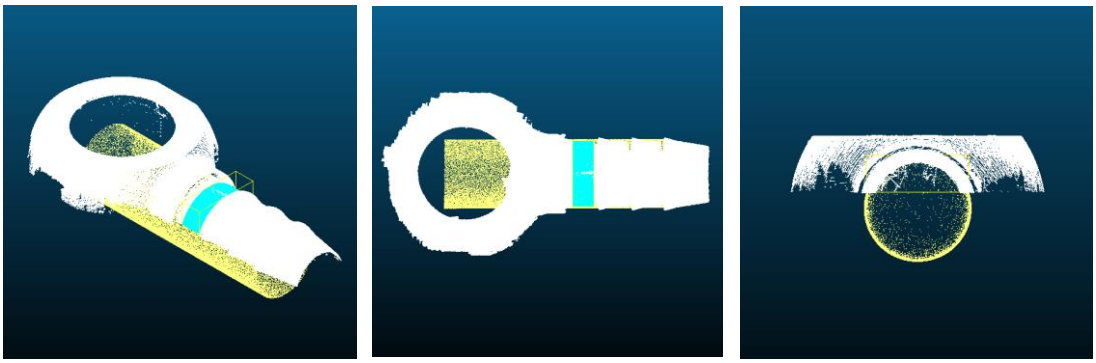
$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)}, \tag{4.10}$$

kde p je pravděpodobnost úspěšného výsledku, ε je pravděpodobnost výběru bodu patřícího mezi *outliers* a s je počet náhodně volených bodů, na jejichž základě výsledný model počítáme. Problémem je však fakt, že parametr ε se bude pro každý segment lišit a jeho hodnotu nelze předem jasně určit.

Algoritmus RANSAC implementovaný v knihovně PCL má několik nastavitelných parametrů, pomocí kterých lze proces ovlivnit a dosáhnout tak co nejlepších výsledků. Dva parametry a jejich význam – počet iterací k a tolerance t – již byly zmíněny. Dalším nezbytným parametrem je model, jehož koeficienty chceme spočítat. Třída SegmentClass podporuje pouze model roviny a válce, které jsou v knihovně definovány pomocí proměnných pcl::SACMODEL_PLANE a pcl::SACMODEL_CYLINDER. Pomocnými argumenty algoritmu, které mají restriktivní vliv na výpočet parametrů modelu, jsou vektor určující výslednou orientaci a maximální úhlovou odchylku ϵ modelu od toho vektoru (normála roviny, vektor určující směr osy válce). Pro výpočet válce je možné ještě nastavit interval hodnot, kterých může nabývat poloměru r. Jednotky parametrů tolerance t a intervalu poloměru r jsou shodné s jednotkami point cloudu. Úhlová

odchylka ε je implementována v PCL knihovně v radiánech, ale metoda SegmentClass:: setEpsilonAngle(double aAngle), která slouží k nastavení této odchylky, předpokládá hodnotu parametru ve stupních a zajišťuje přepočet na radiány.

Výsledkem výpočtu náhradního modelu jsou indexy bodů vstupního point cloudu, které odpovídají vypočtenému modelu (*inliers*) a koeficienty modelu. Model roviny má čtyři koeficienty, které odpovídají parametrům *a*, *b*, *c* a *d* z obecné rovnice roviny (4.4). V případě válce je vráceno sedm koeficientů – první tři jsou souřadnice [x, y, z] bodu, který, se nachází na ose válce, další tři jsou složky (u_x, u_y, u_z) vektoru definujícího směr osy válce a posledním koeficientem je poloměr válce [13]. Příklad vypočteného náhradního modelu válce pro odpovídající segment je zobrazen na obrázku 4.10.



Obrázek 4.10 Výpočet náhradního modelu válce (žlutý) pro daný segment (modrý) naskenovaného dílu pomocí metody RANSAC

4.2.8 Convenient_functions.cpp

V tomto souboru se nachází několik pomocných výpočetních funkcí, které jsou využívány na různých místech programu. Mezi tyto funkce patří například výpočet skalárního součinu dvou vektorů, úhlu mezi dvěma vektory, vzdálenosti bodu od normálového vektoru roviny atd.

4.2.9 Segment_deviation.cpp

Funkce Segment_deviation zajišťuje měření intervalu odchylky všech bodů segmentu od jeho náhradního modelu. Akceptovanými náhradními modely jsou válec a rovina, a proto může být vypočtený interval odchylky považován za válcovitost nebo rovinnost. Uživatel si může vybrat jednu z pěti níže zmíněných variant této funkce. Jednotlivé varianty se liší v množství vypočtených parametrů. Význam jednotlivých parametrů je vysvětlen přímo v programu.

```
int SegmentDeviation(SegmentClass& object1, float& aDeviation);
int SegmentDeviation(SegmentClass& object1, float& aDeviation, float&
                     aMaxDeviation, float& aMinDeviation);
int SegmentDeviation(SegmentClass& object1, const float aTolerance,
                     float& aDeviation, bool& aInTolerance);
```

```
int SegmentDeviation(SegmentClass& object1, const float aTolerance,
                     float& aDeviation, bool& aInTolerance, float&
                     aMaxDeviation, float& aMinDeviation);
int SegmentDeviation(SegmentClass& object1,
                     const float aTolerance, float& aDeviation,
                     bool& aInTolerance, float& aMaxDeviation,
                     float& aMinDeviation, float& aMeanDeviation);
```

Pokud předaný objekt obsahuje rovinný segment, vzdálenost bodů od náhradního modelu roviny je vypočtena podle rovnice

$$l = \frac{|a \cdot x + b \cdot y + c \cdot z + d|}{\sqrt{a^2 + b^2 + c^2}}, \tag{4.11}$$

kde l je vzdálenost bodu od roviny a, b, c, d jsou koeficienty modelu roviny vektoru roviny a x, y, z jsou souřadnice měřeného bodu.

Model válce je reprezentován bodem na ose válce, vektorem určujícím směr osy a poloměrem válce. Pro výpočet vzdálenosti bodu od osy válce je použita rovnice

$$l = \sqrt{|\vec{v}|^2 - \left(\frac{\vec{u} \cdot \vec{v}}{\vec{u}}\right)^2}, \tag{4.12}$$

kde l je vzdálenost bodu od osy válce, $\vec{u}$ je vektor osy válce a $\vec{v}$ je vektor směřující z referenčního bodu na ose válce do měřeného bodu. Nakonec je vypočten rozdíl mezi vzdáleností l a poloměrem r válce náhradního modelu.

Výpočtem maximální vzdálenosti bodu od roviny nebo v případě válce maximální a minimální odchylky vzdálenosti bodu od ideálního poloměru r můžeme stanovit rovinnost, resp. válcovitost daného segmentu.

4.2.10 Segment_distance.cpp

Tato funkce, která slouží k měření vzdálenosti dvou rovinných segmentů, je připravena ve dvou variantách, které ukazuje níže uvedený úryvek kódu. Obě varianty se liší pouze v tom, že první předpis navíc vyžaduje stanovení ideální měřené vzdálenosti, tolerance této vzdálenosti a proměnnou, do které je navrácen příznak, jestli se změřená vzdálenost nachází v této toleranci. Druhá varianta slouží pouze ke změření vzdálenosti mezi rovinami a nevyhodnocuje, zda je vzdálenost v toleranci.

```
int SegmentDistance(SegmentClass& object1, SegmentClass& object2,
                    float aIdealDistance, float aTolerance,
                    double& aMeasuredDistance, bool& aInTolerance);

int SegmentDistance(SegmentClass& object1, SegmentClass& object2,
                    double& aMeasuredDistance);
```

Měření vzdálenosti dvou rovin dává smysl jen pokud jsou roviny rovnoběžné. I kdyby byly dvě plochy skenovaného dílu rovnoběžné (s nekonečnou přesností), proces

skenování, kdy může například už jen vlivem kvantifikace docházet k odchylkám, způsobí, že vypočtený náhradní model roviny bude mít malou úhlovou odchylku od skutečné plochy dílu. Proto je měření vzdálenosti dvou ploch převedeno na měření průměrné vzdálenosti bodů jednoho segmentu od náhradního modelu roviny druhého segmentu. Každý ze segmentů však může, a z pravidla bude, obsahovat různý počet bodů, což by způsobovalo různé výsledky v závislosti na tom, z jakého segmentu by byly brány body a z jakého segmentu by byl použit náhradní model roviny. Proto je proces měření zopakován s tím rozdílem, že ve druhém kroku jsou brány body ze segmentu, ze kterého byl původně brán model roviny a naopak. Výsledná vzdálenost je tedy průměrem vzdáleností bodů segmentu 1 i 2. Výpočet vzdálenosti bodu od roviny popisuje již výše zmíněné rovnice (4.11).

4.2.11 Segment_angle.cpp

Poslední funkcí zajišťující měření geometrických vlastností objektu je Segment_angle. Tato funkce má čtyři varianty, přičemž první dvě porovnávají úhel mezi dvěma segmenty a zbylé dvě varianty slouží k měření úhlu orientace segmentu vůči vektoru zadanému uživatelem. Varianty funkce se liší podle požadovaných výstupních hodnot, které jsou ukládány do proměnných předávaných funkci jako její parametry. Definici jednotlivých variant uvádí následující úryvek kódu.

```
int SegmentAngle(SegmentClass& object1, SegmentClass& object2,
           double aIdealAngle, double aTolerance, double aMeasuredAngle,
           bool& aInTolerance);

int SegmentAngle(SegmentClass& object1, SegmentClass& object2,
           double aMeasuredAngle);

int SegmentAngle(SegmentClass& object1, std::vector<float> aVector,
           double aIdealAngle, double aTolerance,
           double& aMeasuredAngle, bool& aInTolerance);

int SegmentAngle(SegmentClass& object1, std::vector<float> aVector,
           double& aMeasuredAngle);
```

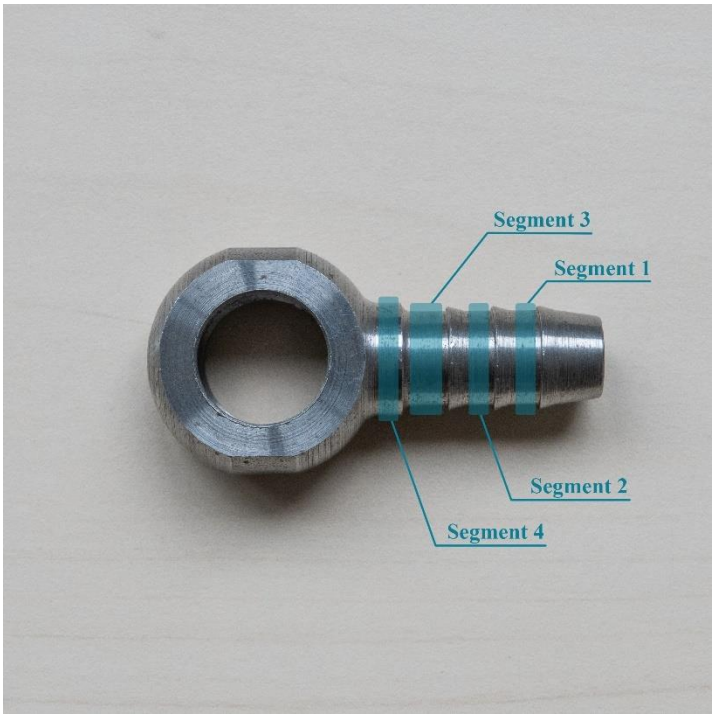
Pro výpočet úhlu využívá funkce koeficienty náhradních modelů jednotlivých segmentů. V případě roviny se jedná o její normálový vektor a v případě válce je použit vektor určující směr osy válce. První varianta funkce slouží k výpočtu úhlu mezi dvěma segmenty, druhá varianta slouží k ověření orientace segmentu vůči ideální požadované orientaci. Samotný výpočet úhlu je proveden na základě rovnice (4.13). Chyba výpočtu úhlu je závislá na přesnosti výpočtu náhradních modelů daného segmentu.

$$\cos\alpha = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}|\cdot|\vec{v}|} \tag{4.13}$$

5. TESTOVACÍ MĚŘENÍ

V této kapitole budou představeny výsledky testů, které měly za úkol otestovat funkčnost navržené sestavy pro automatické skenování konstrukčních dílů za účelem měření jejich geometrických rozměrů a vlastností. Testovací pracoviště bylo popsáno již v kapitole 2.3.

K testování byl použit konstrukční díl, který můžeme vidět na obrázku 2.6. Změřené rozměry pomocí programu bylo potřeba porovnat s reálnými hodnotami testovacího dílu. Byly stanoveny 4 segmenty, u nichž byl pomocí digitálního posuvného měřidla změřen jejich průměr, ze kterého byl následně stanoven poloměr. Tyto segmenty jsou zvýrazněny na obrázku 5.1.



Obrázek 5.1 Měřené segmenty testovacího dílu

K měření bylo použito posuvné měřidlo Mitutoyo CD-15APX s přesností měření ±0,02 mm. Pro každý segment bylo provedeno 10 měření jeho průměru a výsledné hodnoty byly vypočteny podle následujících rovnic.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,, \tag{5.1}$$

$$u_A = k_s \cdot \sqrt{\frac{1}{N \cdot (N-1)} \cdot \sum_{i=0}^{N} (x_i - \bar{x})^2} \,, \tag{5.2}$$

$$u_B = \sqrt{\left(\frac{\Delta_{zmax}}{\chi}\right)^2}, \tag{5.3}$$

$$u_c = \sqrt{{u_a}^2 + {u_b}^2}, \tag{5.4}$$

$$U = u_c \cdot k_r\,, \tag{5.5}$$

kde $\bar{x}$ je aritmetický průměr naměřených hodnot, N je počet měření, u_A je nejistota typu A, k_s je koeficient používaný pro nižší počet měření než N = 10, u_B je nejistota typu B, Δ_{zmax} je maximální odchylka naměřené hodnoty daná měřicím přístrojem, χ je koeficient tvaru rozložení, u_c je kombinovaná nejistota, U je rozšířená nejistota měření a k_r je koeficient rozšíření intervalu. Pro účely veškerých níže provedených měření byly použity koeficienty $\chi = \sqrt{3}$, který odpovídá rovnoměrnému rozložení a $k_r = 2$, což znamená, že skutečná hodnota se nachází uvnitř vypočítaného intervalu daného rozšířenou nejistotou U s 95% pravděpodobností. Naměřené a vypočtené hodnoty nejistot jsou zapsány v tabulce 5.1.

Tabulka 5.1 Rozměry a vypočtené nejistoty testovacího dílu

	$r_{prům}$ [mm]	u_a [mm]	u_b [mm]	u_c [mm]	U [mm]	Výsledný r [mm]
Segment 1	5,257	0,000816	0,0115	0,0116	0,0232	5,257 ± 0,024
Segment 2	5,254	0,000764	0,0115	0,0116	0,0231	5,254 ± 0,024
Segment 3	5,258	0,001333	0,0115	0,0116	0,0232	5,258 ± 0,024
Segment 4	6,238	0,000816	0,0115	0,0116	0,0232	6,238 ± 0,024

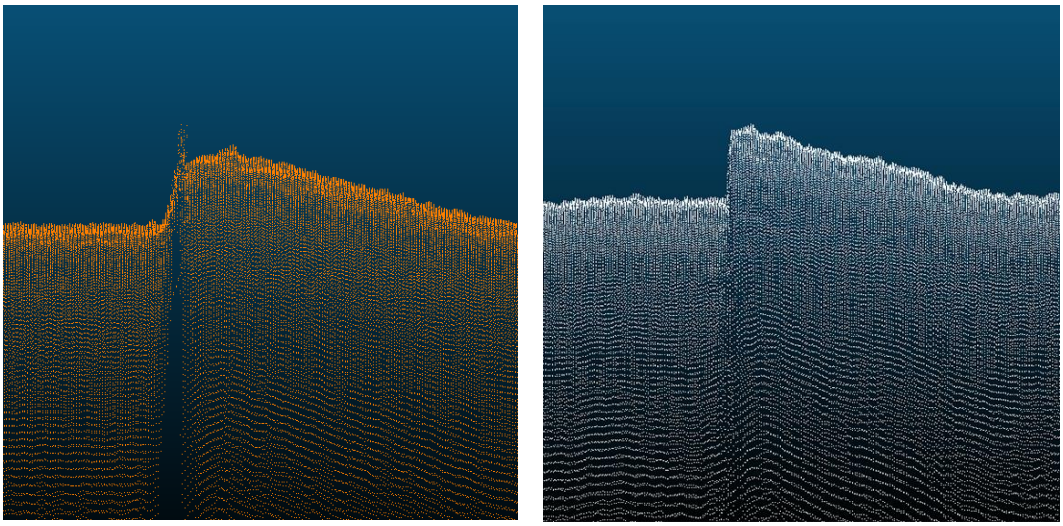
Podle CAD modelu testovacího dílu by segmenty 1, 2 a 3 měly mít poloměr r = 5,3 mm a poloměr segmentu 4 by měl měřit 6,25 mm. Výrobní odchylka rozměrů testovacího dílu je 0,2 mm (pro poloměr je tedy uvažována hodnota 0,1 mm). Změřený poloměr segmentu 4 při uvážení vypočtené nejistoty odpovídá rozměrům CAD modelu. Poloměry segmentů 1, 2 a 3 se od ideální hodnoty liší o méně než 0,02 mm, což je v rámci výrobní tolerance. Takto malá odchylka reálného dílu od CAD modelu by neměla způsobovat žádné problémy ani při zarovnávání point cloudu skenu vůči point cloudu modelu.

Dalším krokem testování bylo pořízení skenů konstrukčního dílu a zpracování naskenovaných dat pomocí navrženého programu. Program kromě poloměru jednotlivých segmentů měřil také jejich válcovitost a úhlovou odchylku jejich os. Parametry skeneru použité při testování byly uvedeny již v kapitole 2.1.1 v tabulce 2.1. Parametry, se kterými byl spouštěn měřicí program, se nachází v tabulce v příloze práce. Pro každé měření bylo pořízeno 10 skenů. Aby mohlo být měření označeno za úspěšné, nesmí se zjištěná hodnota poloměru segmentu lišit o více než 0,01 mm od reálné hodnoty

v tabulce 5.1. Tato hranice úspěšnosti je stanovena jako desetina výrobní tolerance testovacího dílu.

5.1 Vliv mrtvého úhlu skeneru

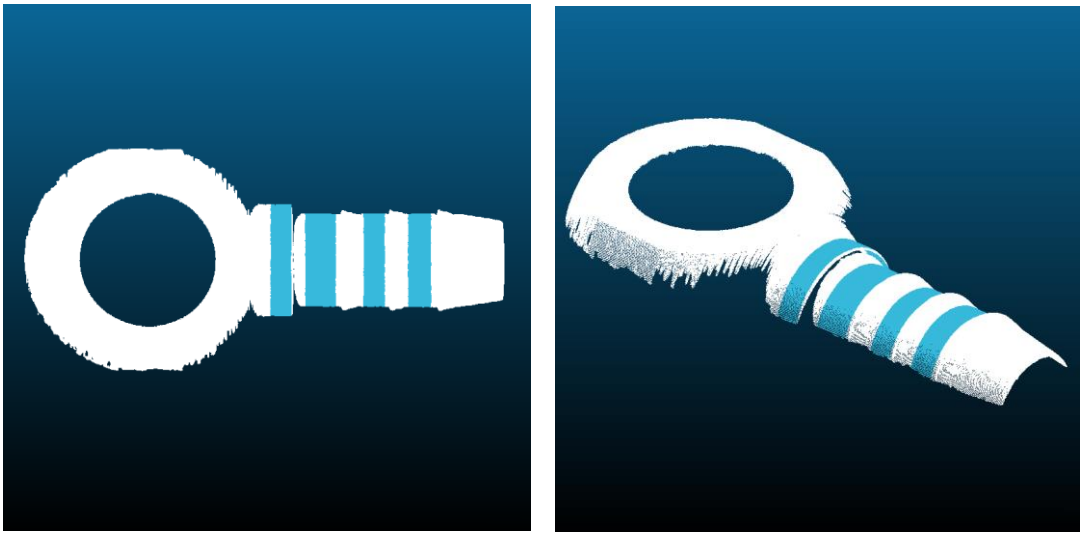
Problematika mrtvého úhlu v souvislosti se skenováním pomocí jednoho skeneru byla již zmíněna v kapitole 2.1. Tento nežádoucí efekt lze pozorovat i při skenování konstrukčního dílu, který byl vybrán pro testovací měření. K minimalizování tohoto efektu stačí, aby byl konstrukční díl vhodně orientovaný vůči skeneru. Efekt mrtvého úhlu je možné pozorovat u vroubkované části testovacího dílu, jak je ukázáno na obrázku 5.2. V případě oranžového point cloudu (obrázek nalevo) byl díl vůči skeneru orientován nevhodně, napravo vidíme bílý point cloud při vhodné orientaci dílu.



Obrázek 5.2 Vliv mrtvého úhlu skeneru – sken testovacího dílu.

5.2 Měření testovacího dílu

Pro měření úhlové odchylky jednotlivých segmentů byl vytvořen referenční segment, který vznikl spojením segmentu 1, 2 a 3. Lze totiž předpokládat, že náhradní model válce spočítaný pro tento referenční segment, bude lépe odpovídat skutečné ose zarovnaného point cloudu skenu. Alternativou by bylo uvažovat jako referenci osu modelu, avšak v tomto případě by se do naměřených odchylek projevovala chyba způsobená nedokonalým zarovnáním skenu vůči modelu. Pro úplnost je vyhodnocována i úhlová odchylka referenčního segmentu od ideální osy modelu. Výsledky měření by měly vypovídat také o opakovatelnosti měřicího systému, což je vyhodnocováno na základě výpočtu nejistoty typu A. Skutečné segmenty vytvořené programem jsou zobrazeny na obrázku 5.3.



Obrázek 5.3 Segmenty vytvořené navrženým programem

5.2.1 Měření se srovnaným dílem

Při tomto měření byl při jednotlivých skenováních testovací díl orientován vždy stejně. K zajištění stálé orientace dílu byla použita speciálně tvarovaná podložka vyrobená na 3D tiskárně. Naměřené hodnoty byly zpracovány podle rovnic (5.1) – (5.5). Výsledky měření se nachází v tabulce 5.2.

Tabulka 5.2 Výsledky měření se srovnaným dílem

	r [mm]	**u_a - r [mm]**	**α [°]**	**u_a - α [°]**	**Válcovitost [mm]**	**u_a - válc. [mm]**
Segment 1	5,064 ± 0,019	0,009	0,69 ± 0,16	0,076	0,119 ± 0,010	0,005
Segment 2	5,065 ± 0,019	0,009	0,84 ± 0,26	0,129	0,138 ± 0,024	0,012
Segment 3	5,072 ± 0,021	0,010	0,65 ± 0,23	0,114	0,158 ± 0,029	0,014
Segment 4	6,016 ± 0,009	0,004	0,49 ± 0,15	0,075	0,169 ± 0,024	0,012
Segment ref	5,063 ± 0,026	0,013	1,0 ± 0,7	0,310	0,174 ± 0,019	0,009

5.2.2 Měření s rozdílnou orientací dílu

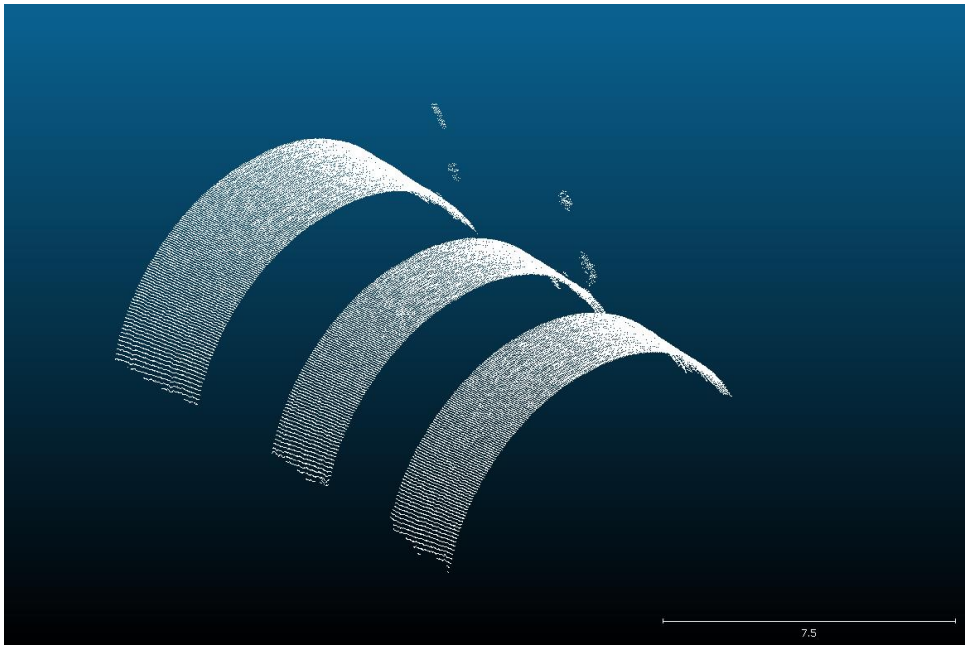
Cílem měření s rozdílnou orientací dílu bylo ověřit robustnost zarovnávacího algoritmu implementovaného v navrženém programu. Testovací díl je proto mezi jednotlivými měřeními náhodně otáčen kolem osy z, maximální výchylka od srovnané polohy z předchozího případu je přibližně 40° na každou stranu. Omezení natočení je dáno nejdelším rozměrem dílu a maximální šířkou profilu skeneru. Naměřené hodnoty byly zpracovány stejným způsobem jako při měření se srovnaným dílem. Výsledné hodnoty se nachází v následující tabulce 5.3.

Tabulka 5.3 Výsledky měření s rozdílnou orientací dílu

	r [mm]	u_a - r [mm]	α [°]	u_a - α [°]	Válcovitost [mm]	u_a - válc. [mm]
Segment 1	5,107 ± 0,018	0,009	1,1 ± 0,5	0,217	0,6 ± 0,5	0,225
Segment 2	5,068 ± 0,016	0,008	0,8 ± 0,4	0,151	0,7 ± 0,6	0,253
Segment 3	5,065 ± 0,025	0,012	0,63 ± 0,27	0,131	0,9 ± 0,6	0,287
Segment 4	6,037 ± 0,027	0,013	1,17 ± 0,26	0,127	0,216 ± 0,029	0,014
Segment ref	5,08 ± 0,05	0,021	0,61 ± 0,17	0,084	1,0 ± 0,6	0,284

5.2.3 Vyhodnocení změřených hodnot

V obou případech měření zvládl program vypočítat všechny požadované hodnoty. Při porovnání hodnot, které se nachází v tabulce 5.2 a 5.3, vidíme, že výsledky měření se velice podobají. Na základě těchto dvou faktů lze konstatovat, že zarovnávací a měřicí algoritmy implementované v programu jsou dostatečně robustní. Jediný větší rozdíl lze pozorovat u válcovitosti jednotlivých segmentů. Při měření s rozdílnou orientací dílu mohlo při jistém natočení docházet ke zvýšenému výskytu nežádoucích odrazů a odlesků laserového paprsku, a tudíž ke vzniku většího množství falešných bodů v point cloudu skenu, které se nepodařilo odstranit ani použitým filtrem StatisticalOutlierRemoval. Jeden takový point cloud obsahující velké množství falešných bodů představuje obrázek 5.4.

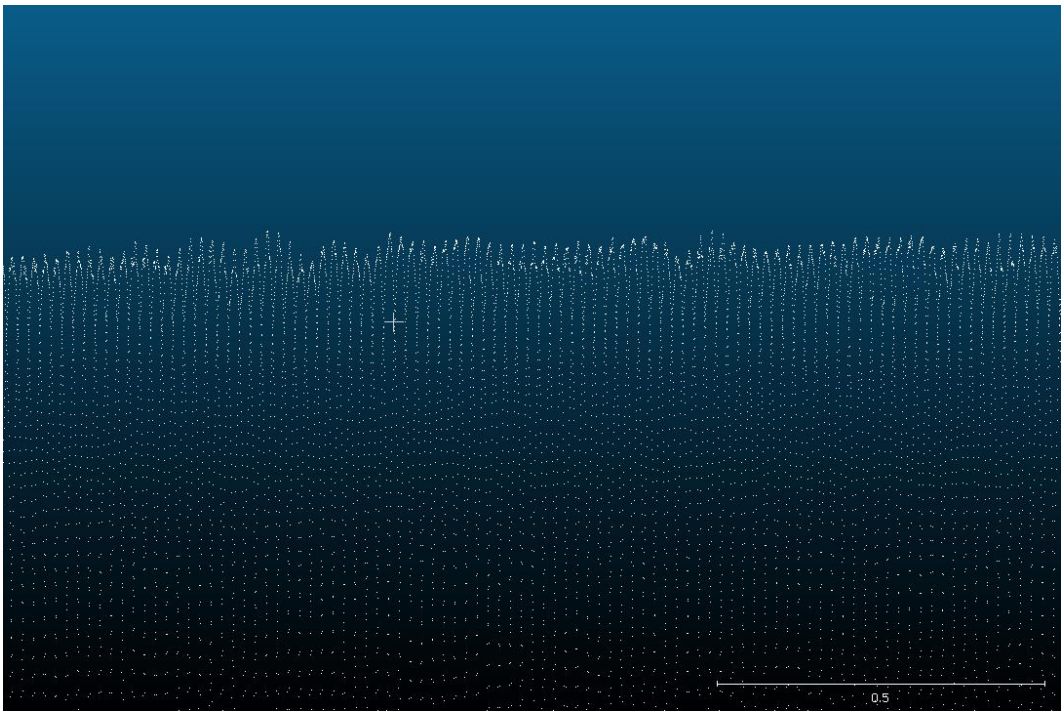


Obrázek 5.4 Falešné body nacházející se v měřených segmentech

Při porovnání výsledků měření z tabulky 5.2 a 5.3 s reálnými hodnotami v tabulce 5.1 bylo zjištěno, že poloměry segmentů 1 – 4 měřené pomocí skeneru se liší od reálných hodnot přibližně o 0,2 mm. Stejně tak změřené úhlové odchylky a válcovitosti nabývají

poměrně vysokých hodnot. Pozitivně lze hodnotit opakovatelnost měření poloměru hodnocenou podle nejistoty typu A, která se při obou měřeních pohybovala kolem hodnoty 0,01 mm.

Při detailním bočním pohledu na měřené segmenty point cloudu skenu lze pozorovat znatelné zvlnění povrchu (viz obrázek 5.5). Rozptyl výšky jednotlivých bodů neboli jejich souřadnice v ose z dosahuje hodnot kolem 0,05 mm. Tato nerovnost povrchu může mít negativní vliv na výpočet náhradního modelu daného segmentu. Při pohledu na skutečný testovací díl je na jeho povrchu možné pouhým okem pozorovat rýhování, které pravděpodobně vzniklo během obrábění tohoto dílu. Řešením by mohlo být vyhlazení naskenovaných dat pomocí mediánového filtru, který pomáhá potlačit extrémní hodnoty v rámci zvoleného okolí. Tato funkce lze nastavit přímo v kontroléru skeneru.



Obrázek 5.5 Boční detailní pohled na měřený segment

5.3 Měření s mediánovým filtrem

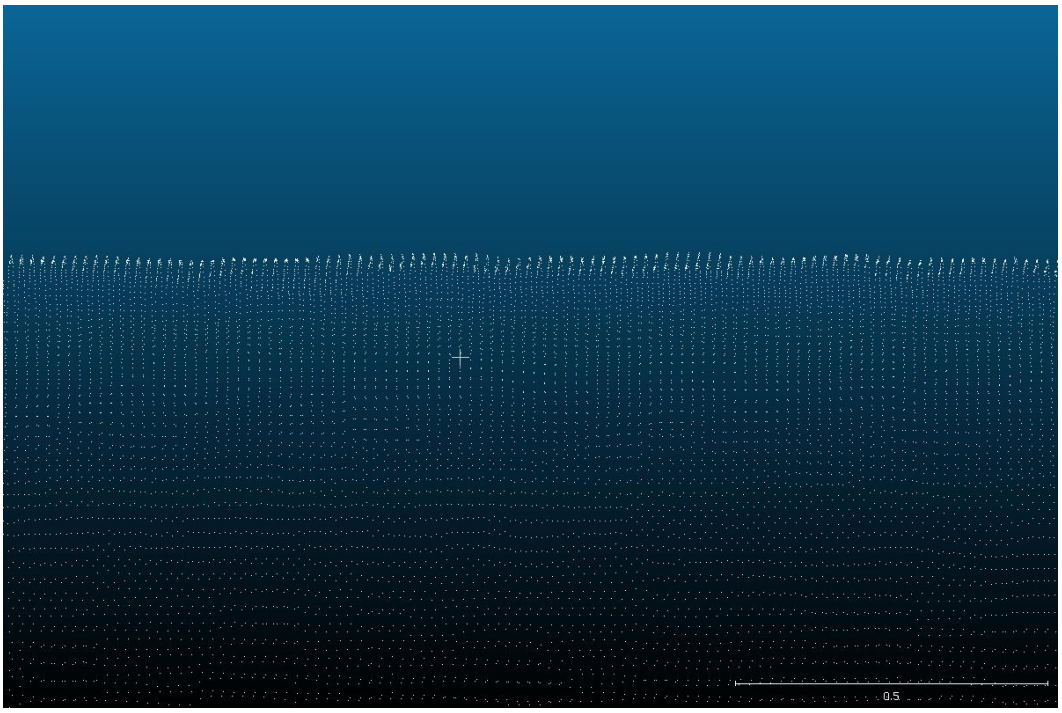
Cílem tohoto měření bylo ověření teorie, jestli použití mediánového filtru na naskenované hodnoty nepovede k lepším výsledkům měření. Měření bylo provedeno stejně jako v předchozím případě, jediná změna spočívala ve změně nastavení řídicí jednotky skeneru. Pro toto měření byl přidán mediánový filtr souřadnic osy z o velikosti 7 (v tabulce 2.1 je tento parametr pojmenován jako Filter process time-axis median). Rozdíl mezi mediánovým filtrem pro osu x a pro osu z (časová osa) je, že mediánový filtr osy x bere v potaz pouze hodnoty v rámci jednoho profilu. Druhá zmíněná varianta filtru pracuje naopak s hodnotami odpovídajících bodů napříč zvolným počtem profilů.

Naměřené hodnoty jsou zpracovány stejným způsobem jako v předchozím případě. Výsledky měření jsou zapsány v tabulce 5.4.

Tabulka 5.4 Výsledky měření se srovnaným dílem a mediánovým filtrem

	r [mm]	**u_a - r [mm]**	**α [°]**	**u_a - α [°]**	**Válcovitost [mm]**	**u_a - válc. [mm]**
Segment 1	5,056 ± 0,022	0,011	0,76 ± 0,25	0,121	0,109 ± 0,024	0,012
Segment 2	5,071 ± 0,008	0,004	0,45 ± 0,16	0,076	0,074 ± 0,007	0,003
Segment 3	5,077 ± 0,016	0,008	0,29 ± 0,05	0,021	0,149 ± 0,020	0,010
Segment 4	6,025 ± 0,015	0,007	0,64 ± 0,22	0,108	0,095 ± 0,007	0,003
Segment ref	5,071 ± 0,020	0,010	0,30 ± 0,08	0,039	0,153 ± 0,020	0,010

Při porovnání výsledků měření s mediánovým filtrem a výsledků z tabulky 5.2 můžeme pozorovat zlepšení v úhlové odchylce referenčního segmentu od ideální osy modelu. Úhlové odchylky segmentu 2 a 3 se sice zlepšily, ale u segmentu 1 a 4 došlo naopak k mírnému zhoršení, takže nelze konstatovat celkové zlepšení výsledku. Taktéž změřené poloměry všech segmentů vyšly téměř identicky jako při měření bez mediánového filtru. Použití mediánového filtru sice vyřešilo zkřivení povrchu testovacího dílu (viz obrázek 5.6), lepších výsledků měření však nebylo dosaženo. Důvod nepřesnosti měření se nepodařilo určit.



Obrázek 5.6 Boční detailní pohled na měřený segment při použití mediánového filtru

5.4 Ověření funkčnosti programu

Tato část se věnuje ověření funkčnosti programu zajišťujícího zpracování a vyhodnocování naskenovaných dat. Hodnoty získané ze skenů pořízených pomocí skenovací sady navržené v kapitole 2 vykazovaly nežádoucí chyby, a proto bylo provedeno kontrolní měření na ideálních datech CAD modelu testovacího dílu, aby byla ověřena správná funkčnost programu.

Podle CAD modelu měl být poloměr segmentu 1, 2 a 3 roven 5,3 mm a poloměr segmentu 4 měl být 6,25 mm. V tomto případě, kdy byla známá přesná orientace modelu v prostoru, byla úhlová odchylka jednotlivých segmentů měřena vůči ideálnímu vektoru, který byl shodný s osou x, a nikoliv vůči ose náhradního modelu referenčního segmentu. Válcovitost v tomto případě nebyla počítána, neboť CAD model obsahoval i vnitřní stěny dílu, které by měření válcovitosti zkreslovaly. Vzhledem k tomu, že se jednalo o práci s ideálními daty a nebyl k dispozici více než jeden CAD model testovacího dílu, bylo provedeno pouze jedno měření. Při opakování procesu měření na datech modelu byly získány pokaždé stejné výsledky. Pro ověření vlivu hustoty point cloudu byly z CAD modelu vygenerovány tři point cloudy o různé hustotě bodů. První model M1 byl tvořen 250 tisíci body, model M2 tvořilo 500 tisíc bodů a model M3 byl tvořen 2,5 miliony body. Množství bodů pro jednotlivé point cloudy bylo voleno s ohledem na fakt, že point cloud skenu je tvořen přibližně 660 tisíci body, které však reprezentují v porovnání s modelem menší část dílu. Jednotlivé point cloudy modelu jsou zobrazeny na obrázku 5.7. Hodnoty vypočtené jsou zapsány v tabulce 5.5.

Tabulka 5.5 Výsledky měření se srovnaným dílem a mediánovým filtrem

	r_{M1} [mm]	α_{M1} [°]	r_{M2} [mm]	α_{M2} [°]	r_{M3} [mm]	α_{M3} [°]
Segment 1	5,30	0,13	5,27	0,01	5,27	0,05
Segment 2	5,29	0,05	5,27	0,09	5,27	0,05
Segment 3	5,29	0,03	5,29	0,56	5,29	0,04
Segment 4	6,23	0,03	6,23	0,02	6,22	0,22
Segment ref	5,28	0,01	5,27	0,04	5,30	0,05

Naměřené poloměry všech segmentů nevykazují odchylku od správné hodnoty větší něž 0,03 mm. Ve většině případů se i naměřená úhlová odchylka pohybuje v řádu pouze několika setin stupňů. Z výše zjištěných výsledků lze také vyvodit, že ani zvyšující se hustota bodů nemá na změřené hodnoty zásadní vliv. Velmi přesných výsledků bylo dosaženo i v případě point cloudu M1, jehož hustota bodů je několikanásobně nižší než hustota point cloudu skenu z předchozích měření.

Obrázek 5.7 Testovací point cloudy modelu s různou hustotou bodů. Z leva point cloud M1, M2 a M3

5.5 Testování na externích datech

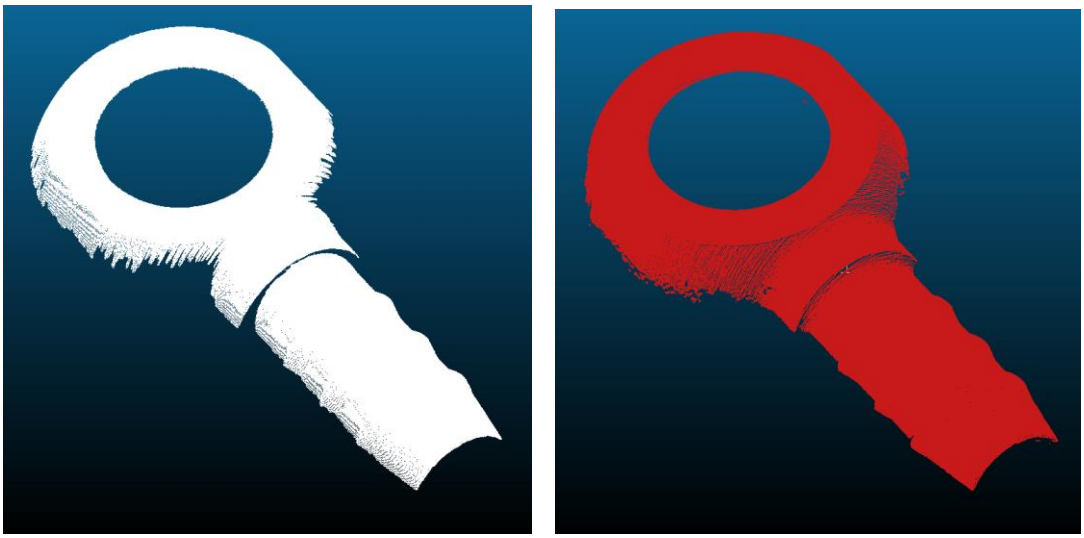
K poslednímu testovacímu měření, provedenému v rámci této práce, byla využita data, která byla dodána firmou Acam Solution. Tato data byla pořízena skenerem Keyence LJ-X8080 řízeného kontrolérem LJ-X8000, ostatní použité periferie a nastavení kontroléru nebyly známé. Jedná se o vyšší řadu skenerů, než byla zapůjčena pro realizaci měřicího stanoviště v rámci této práce. Hlavní výhodou skeneru LJ-X8080 oproti modelu LJ-V7080 je větší rozlišení bodů v rámci jednoho profilu (3200 versus 800). Pořízené skeny by tak mohly být kvalitnější a mohlo by tak být docíleno přesnějšího měření. Měření bylo provedeno na 9 skenech (tento fakt byl zohledněn při výpočtu nejistoty typu A), přičemž testovací díl během skenování nebyl nijak aretován do stálé pozice.

Tabulka 5.6 Výsledky měření externích dat

	r [mm]	**u_a - r [mm]**	**α [°]**	**u_a - α [°]**	**Válcovitost [mm]**	**u_a - válc. [mm]**
Segment 1	5,24 ± 0,08	0,039	0,6 ± 0,4	0,164	0,3 ± 0,4	0,164
Segment 2	5,25 ± 0,08	0,040	1,1 ± 1,6	0,794	0,3 ± 0,4	0,169
Segment 3	5,21 ± 0,06	0,027	0,9 ± 0,9	0,403	0,6 ± 0,8	0,353
Segment 4	6,18 ± 0,04	0,018	0,9 ± 0,5	0,233	0,4 ± 0,6	0,286
Segment ref	5,21 ± 0,04	0,017	0,32 ± 0,19	0,093	0,6 ± 0,8	0,370

Při porovnání naměřených hodnot poloměrů s reálnými hodnotami uvedenými v tabulce (5.1) lze výsledky označit za velmi přesné. Hodnoty úhlových odchylek se příliš neliší od výsledků, které byly získány z vlastních skenů (tabulka 5.2 a 5.3). Jednotlivé nejistoty měření však nabývají v tomto případě vyšších hodnot. Vzhledem k neznalosti podmínek, za jakých byla tato sada skenů pořízena, nelze kvalifikovaně odhadovat zdroj těchto nepřesností. Při porovnání point cloudu skenu získaného pomocí řady LJ-V7000 a druhého pomocí sady LJ-X8000 je vidět (viz obrázek 5.8), že druhý zmíněný point cloud vykazuje o něco lepší pokrytí stěn testovacího dílu, avšak oblasti měřených

segmentů 1 až 4 se v tomto ohledu prakticky neliší. Proto je přesnější výsledek měření poloměru daných segmentů připisován většímu rozlišení point cloudu, který poskytuje řada LJ-X8080.



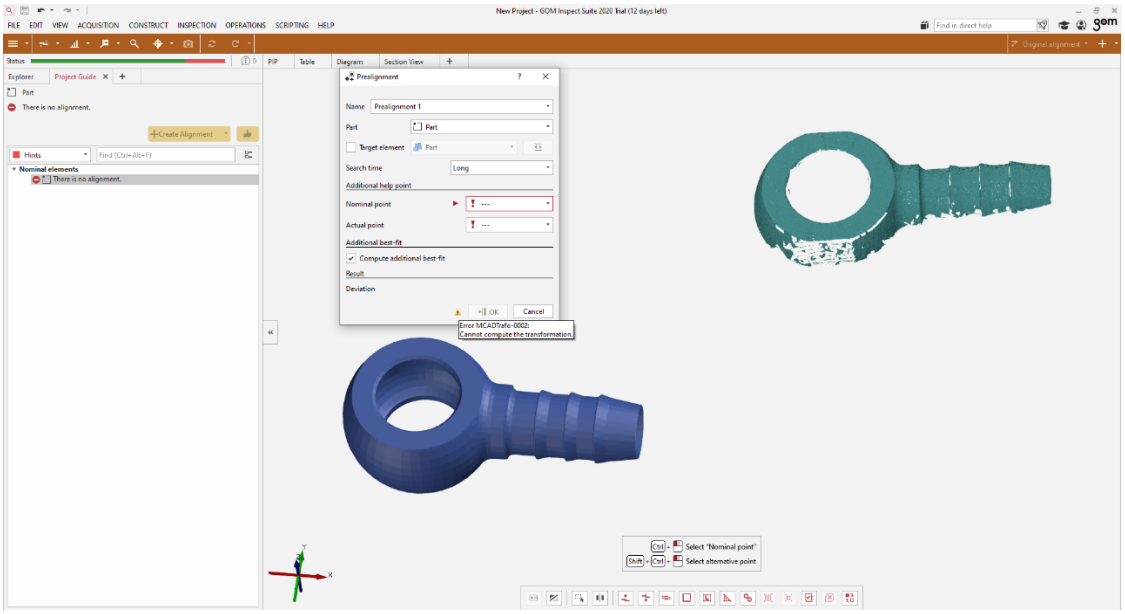
Obrázek 5.8 Nalevo bílý point cloud získaný řadou skeneru LJ-V7000, napravo červený point cloud pořízený řadou LJ-V8000

5.6 GOM Inspect – referenční měření

Na základě rešerše prezentované v kapitole 3 byl program GOM Inspect vybrán jakožto srovnávací softwarový nástroj sloužící k měření rozměrů a geometrických vlastností naskenovaného dílu. K testování byla využita bezplatná 30denní zkušební verze programu.

Proces měření začíná importováním CAD modelu a naskenovaného point cloudu testovacího dílu. Program GOM Inspect funguje na principu měření povrchů jednotlivých objektů vůči sobě. Proto je nutné pro vložený point cloud skenu nejprve spočítat mesh reprezentující jeho povrch. Po vytvoření meshe je potřeba provést zarovnání skenu vůči modelu. U tohoto kroku se však vyskytl problém – GOM Inspect nedokázal sken vůči modelu vůbec zarovnat. Tento problém přetrvával i při různém nastavení procesu vytváření meshe. CAD model a mesh skenu před zarovnáním je zobrazen na obrázku 5.9.

Problém nejspíš pramení z příliš nekvalitního meshe skenu, který navíc reprezentuje pouze malou část testovacího dílu. Pro ověření této teorie byl do programu nahrán point cloud vytvořený z CAD modelu, ke kterému měl být vytvořený point cloud následně zarovnán. S těmito ideálními daty si program poradil bez jakýchkoliv potíží. Problém se zarovnáváním by pravděpodobně mohlo vyřešit, kdyby byl k dispozici sken složený z více dílčích skenů, tak aby byla vytvořena ucelenější a přesnější digitální reprezentace reálného dílu.



Obrázek 5.9 GOM Inspect – problém se zarovnáním skenu vůči modelu

6. ZÁVĚR

Tato diplomová práce si kladla za svůj cíl navrhnout a realizovat automatizovaný systému pro skenování konstrukčních dílů za účelem inspekce jejich rozměrů a geometrických vlastností. Téma bylo navrženo firmou Acam Solution, která by případné funkční řešení mohla využít i v praxi. Firma rovněž zajišťovala veškeré potřebné prostředky pro realizaci projektu.

Úvod práce je věnován teoretickému úvodu do tématu 3D skenování, a to se zaměřením na samotný proces skenování, funkční principy 3D skenerů a na běžně používané typy reprezentací 3D dat.

Následujícím krokem bylo navržení hardwarového uspořádání testovací sestavy. Vzhledem k tomu, že firma nedisponuje vlastním zařízením pro 3D skenování, byl výběr omezen na skenery, které mohly být zapůjčeny od firmy Keyence. Zapůjčená testovací sada se skládala ze skeneru LJ-V7080 a řídicí jednotky LJ-V7001P. Celá sestava byla doplněna o lineární pohon, který zajišťoval pohyb testovacího dílu při skenování a o optické čidlo, které sloužilo k synchronizaci pohonu s jednotkou skeneru. Sestava byla ovládána z PC, na kterém běžel pro tuto práci navržený program zajišťující automatické zpracování pořízených dat.

Před zahájením vývoje programu byla provedena krátká rešerše, která měla za úkol zmapování dostupných softwarových nástrojů určených pro zpracování naskenovaných dat za účelem inspekce jejich rozměrů. Tato rešerše potvrdila předpoklad, že nejvhodnějším nástrojem k realizaci cíle této práce bude knihovna Point Cloud Library.

Hlavním jádrem této práce je návrh programu využívajícím funkce implementované v knihovně PCL. Cílem bylo vytvořit SW nástroj, který automaticky zpracuje naskenovaná data do takové podoby, aby mohly být změřeny rozměry zvolených segmentů a výsledky tohoto měření předány nadřazenému systému v požadované podobě.

Závěrečná fáze práce se věnuje testovacímu měření, které mělo za cíl ověřit použitelnost navrženého řešení. Pro testování byly zvoleny 4 válcovité segmenty testovacího dílu, u nichž byl měřen jejich poloměr, úhlová odchylka a válcovitost. První měření ukázalo, že navržená sestava je sice funkční, avšak nebylo dosaženo dostatečné přesnosti měření. Aby mohlo být měření považováno za dostatečně přesné, musela by být odchylka od skutečné hodnoty poloměru menší než 0,01 mm, změřené hodnoty poloměrů se však lišily přibližně o 0,2 mm. Rovněž naměřené úhlové odchylky byly příliš vysoké. K zásadnímu zlepšení výsledků nedošlo ani při zpracování naskenovaných dat pomocí mediánového filtru, který měl potlačit nežádoucí nerovnosti povrchu testovacího dílu.

Při testování funkčnosti programu na ideálních datech CAD modelu testovacího dílu bylo však dosaženo velice dobré přesnosti ve všech měřených parametrech. Výrazně lepších výsledků bylo dosaženo i při měření dat získaných pomocí vyšší řady skeneru než té, která byla využita pro realizaci skenovací sestavy v rámci této práce. Předpokládá se,

že zlepšení je způsobeno vyšším rozlišení dat pořízených skenerem vyšší řady. Avšak i tyto hodnoty vykazovaly v porovnání s měřením na ideálních datech CAD modelu poměrně vysoké chyby při měření úhlové odchylky jednotlivých segmentů.

Na základě provedených testů lze tvrdit, že v rámci práce byl navržen funkční program splňující požadavek automatické inspekce rozměrů a geometrických vlastností konstrukčního dílu na základě dat pořízených 3D skenerem. Testy však ukázaly, že přesnost změřených výsledků je značně závislá na kvalitě vstupních dat. Kýžené přesnosti by mohlo být dosaženo, pokud by program pracoval s daty reprezentujícími větší část reálného dílu než sken pořízený pouze z jednoho pozorovacího úhlu. Situace by mohla být vyřešena postupným skládáním dílčích skenů, čímž by byla vytvořena dokonalejší digitální reprezentace reálného dílu. Tato varianta by však byla výpočetně i časově náročná, což by komplikovalo případnou implementaci do nadřazeného automatizovaného systému. Vhodnějším řešením by bylo využití funkce řídicí jednotky skeneru, která umí sloučit data ze dvou připojených skenerů již během skenování.

Navržený program a znalosti získané během realizace této práce poslouží firmě Acam Solution při dalším vývoji automatizovaného systému měření rozměrů konstrukčních dílů využívajícím technologii 3D skenování.

LITERATURA

[1] BELLOCCHIO, Francesco, N. Alberto BORGHESE, Stefano FERRARI a Vincenzo PIURI. *3D Surface Reconstruction, Multi-Scale Hierarchical Approaches*. 2013.

[2] BI, Z.M. a Lihui WANG. Advances in 3Ddata acquisition and processing for industrial applications. *Robotics and Computer-Integrated Manufacturing*. 2010, , 403–413.

[3] HSIEH, Cheng-Tiao. An efficient development of 3D surface registration by Point Cloud Library (PCL). In: *International Symposium on Intelligent Signal Processing and Communication Systems*. 2012.

[4] BESL, J.P. a Neil. D MCKAY. A method for registration of 3-D shapes. *Transactions on pattern analysis and machine intelligence* . 1992, **14**(02).

[5] GEBRINO, Salvatore, Domenico MARIA DEL GIUDICE, Gabriele STAIANO, Antonio LANZOTTI a Massimo MARTORELLI. *On the influence of scanning factors on the laser scanner-based 3D*. 2015. Dostupné z: doi:10.1007/s00170-015-7830-7

[6] PRIETO, Flavio, Richard LEPAGE, Pierre BOULANGER a Tanneguy REDARCE. A CAD-based 3D data acquisition strategy for inspection. *Machine Vision and Applications*. 2003. Dostupné z: doi:DOI 10.1007/s00138-003-0131-4

[7] Https://www.nikonmetrology.com/images/brochures/altera-en.pdf. *Nikon Metrology*. 2020.

[8] BOEHLER, Wolfgang a Andreas MARBS. 3D scanning instruments. *Proceedings of the CIPA WGProceedings of the CIPA WG*. 2002.

[9] MCHENRY, Kenton a Peter BAJCSY. *An Overview of 3D Data Content, File Formats and Viewers*. 2008. University of Illinois at Urbana-Champaign, Urbana, IL.

[10] CORPORATION, KEYNCE. *High-speed 2D/3D Laser Scanner LJ-V7000 Series*. 2021.

[11] *LJ-V7000 Series Technology Guide*. KEYENCE CORPORATION, 2020.

[12] RUSU, Radu a Steve COUSINS. 3D is here: Point Cloud Library (PCL). In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, 2011.

[13] Https://pointclouds.org/. *Point Cloud Library*. 2020.

[14] Https://www.gom-inspect.com/. *GOM Inspect*. 2020.

[15] Https://www.3dsystems.com/software/geomagic-control-x. *3D Systems - Geomagic Control X*. 2020.

[16] Https://www.mvtec.com/products/halcon. *MVTec HALCON*. 2020.

[17] RUSU, Radu, Zoltan MARTON, Nico BOLDOW, Mihai DOLHA a Michael BEETZ. *Towards 3D Point Cloud Based Object Maps for*. Mnichov: Technische Universitat M¨ unchen, Computer Science Department, 2010.

[18] MIANO, John. *Compressed image file formats" JPEG, PNG, GIF, XBM, BMP*. 2. ACM Press, 2000.

[19] RUSU, Radu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. Mnichov, 2009. Disertační práce. Technischen Universität München.

[20] TOMBARI, Federico, Samuele SALTI a Luigi DI STEFANO. Unique Signatures of Histograms for Local Surface. In: *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision*. Heraclion, 2010, s. 356-369.

[21] CHUM, Ondřej a Jiří MATAS. Optimal Randomized RANSAC. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2008, , s. 1472 - 1482.

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FEKT	Fakulta elektrotechniky a komunikačních technologií
VUT	Vysoké učení technické v Brně
PCL	Point Cloud Library
CMM	Coordinate Measuring Machine
CCD	Charge-coupled Device
CSD	Constructive Solid Geometry
NURBS	Non-uniform rational basis spline
CAD	Computer-aided Design
ICP	Iterative Closest Point
LRF	Local Reference Frame
SHOT	Signature of Histograms of OrienTations
RANSAC	Random sample consensus

Symboly:

v	rychlost	(m/s)
f	frekvence	(Hz)
l	vzdálenost	(m)

SEZNAM PŘÍLOH

Příloha A - Datový list LJ-V7080

Datový list

KEYENCE

LJ-V7080

Hlava senzoru

CE

SPECIFIKACE

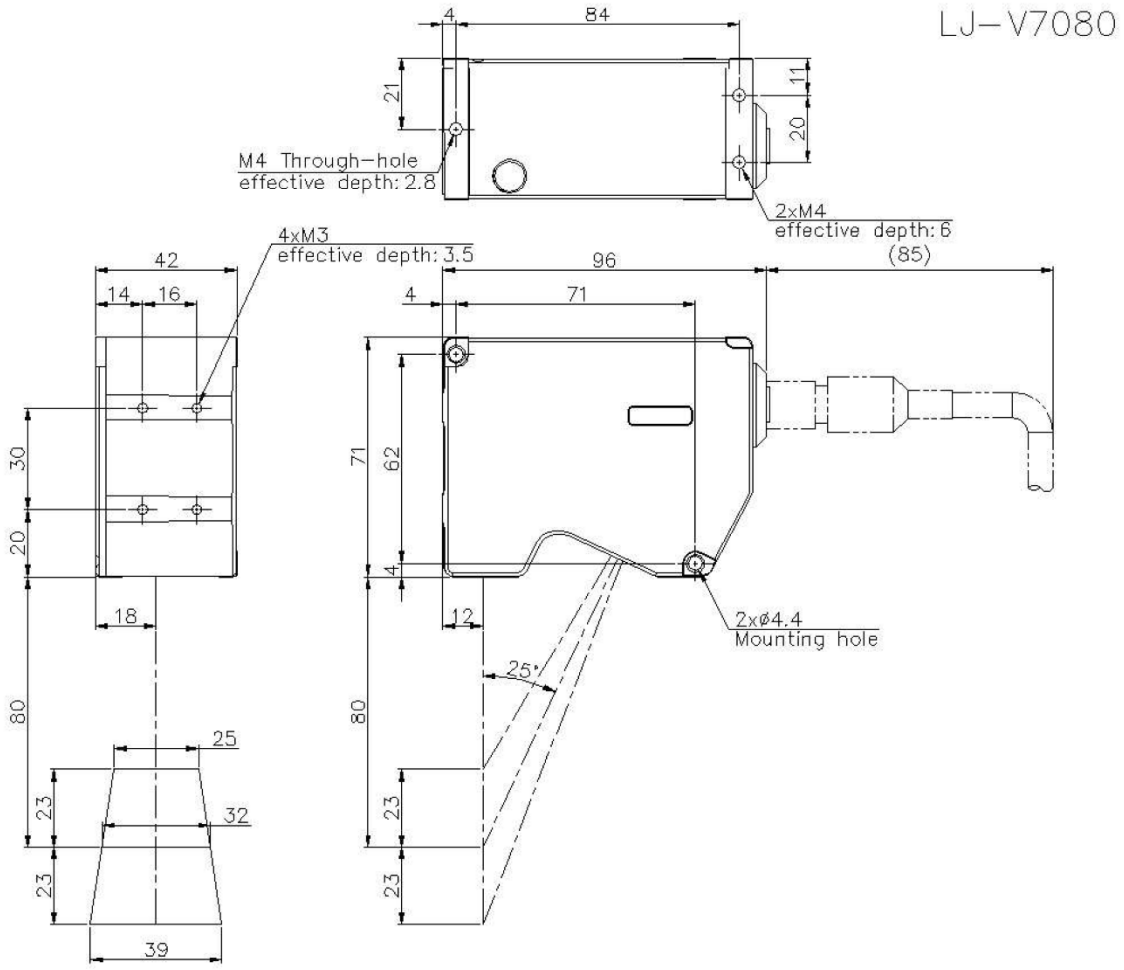
Model			LJ-V7080
Montážní podmínky			Difuzní odraz
Referenční vzdálenost			80 mm
Rozsah měření	Osa Z (výška)		±23 mm (pl. rozs.=46 mm)
	Osa X (šířka)	Bližší strana	25 mm
		Referenční vzdálenost	32 mm
		Vzdálenější strana	39 mm
Světelný zdroj	Typ		Modrý polovodičový laser
	Vlnová délka		405 nm (viditelný paprsek)
	Třída laseru		Laserový produkt třídy 2 (IEC60825-1, FDA (CDRH) část 1040.10[*1])
	Výkon		4,8 mW
Velikost paprsku (referenční vzdálenost)			Cca 48 mm x 48 µm
Opakovatelnost	Osa Z (výška)		0,5 µm[*2*3]
	Osa X (šířka)		10 µm[*4*5]
Linearita	Osa Z (výška)		±0,1 % z pl. rozs.[*6]
Interval bodů profilu	Osa X (šířka)		50 µm
Vzorkovací frekvence (interval spouště)			Nejvyšší rychlost: 16 µs (vysokorychlostní režim), nejvyšší rychlost: 32 µs (pokročilý funkční režim)[*7]
Teplotní charakteristika			0,01 % z pl. rozs./°C
Odolnost vůči prostředí	Stupeň krytí		IP67 (IEC60529)[*8]
	Okolní světlo		Žárovka: max. 10 000 lx[*9]
	Okolní teplota		0 až 45 °C[*10]
	Relativní vlhkost		20 až 85 % RV (bez kondenzace)
	Odolnost vůči vibracím		10 až 57 Hz, dvojitá amplituda 1,5 mm, 3 hodiny ve směru každé z os X, Y a Z
	Odolnost proti nárazu		15 G / 6 ms
Materiál			Hliník
Hmotnost			Cca 400 g

[*1] Klasifikace laseru pro FDA (CDRH) se provádí na základě normy IEC60825-1 v souladu s pokyny pro laserová zařízení (Laser Notice No.50).
[*2] Hodnota vychází ze situace, kdy bylo měření prováděno v referenční vzdálenosti se 4096násobným zprůměrováním.
[*3] Cíle měření jsou standardní cíle KEYENCE. Tato hodnota vychází ze situace, kdy byla průměrná výška oblasti výchozího nastavení měřena ve výškovém režimu. Všechna další nastavení jsou výchozí.
[*4] Tato hodnota vychází ze situace, kdy bylo měření prováděno v referenční vzdálenosti se 4096násobným zprůměrováním.
[*5] Cíl měření je mikrometrický odpich. Tato hodnota vychází ze situace, kdy byla poloha průsečíku mezi zaobleným povrchem mikrometrického odpichu a úrovní hrany měřena v polohovém režimu. Všechna další nastavení jsou výchozí.
[*6] Cíle měření jsou standardní cíle KEYENCE. Profilová data vycházejí ze situace, kdy bylo měření prováděno se 64násobným vyhlazením a 8násobným zprůměrováním. Všechna další nastavení jsou výchozí.
[*7] V případě vysokorychlostního režimu, kdy je oblast měření na své minimální hodnotě, je zapnuta funkce binning, je nastaven standardní režim záznamu obrazu a je zapnut paralelní záznam obrazu. Všechna ostatní nastavení jsou výchozí. V případě pokročilého funkčního režimu, kdy je oblast měření na své minimální hodnotě, je zapnuta funkce binning a je nastaven standardní režim záznamu obrazu. Všechna další nastavení jsou výchozí.
[*8] Tato hodnota vychází ze situace, kdy byl připojen kabel ke hlavě senzoru (CB-B*) nebo prodlužovací kabel (CB-B*E).
[*9] Toto je osvětlení určené pro povrch hlavy senzoru přijímající světlo v průběhu měření bílého papíru, kdy světlo svítí na bílý papír.
[*10] K použití je vyžadována instalace hlavy senzoru na kovovou desku.

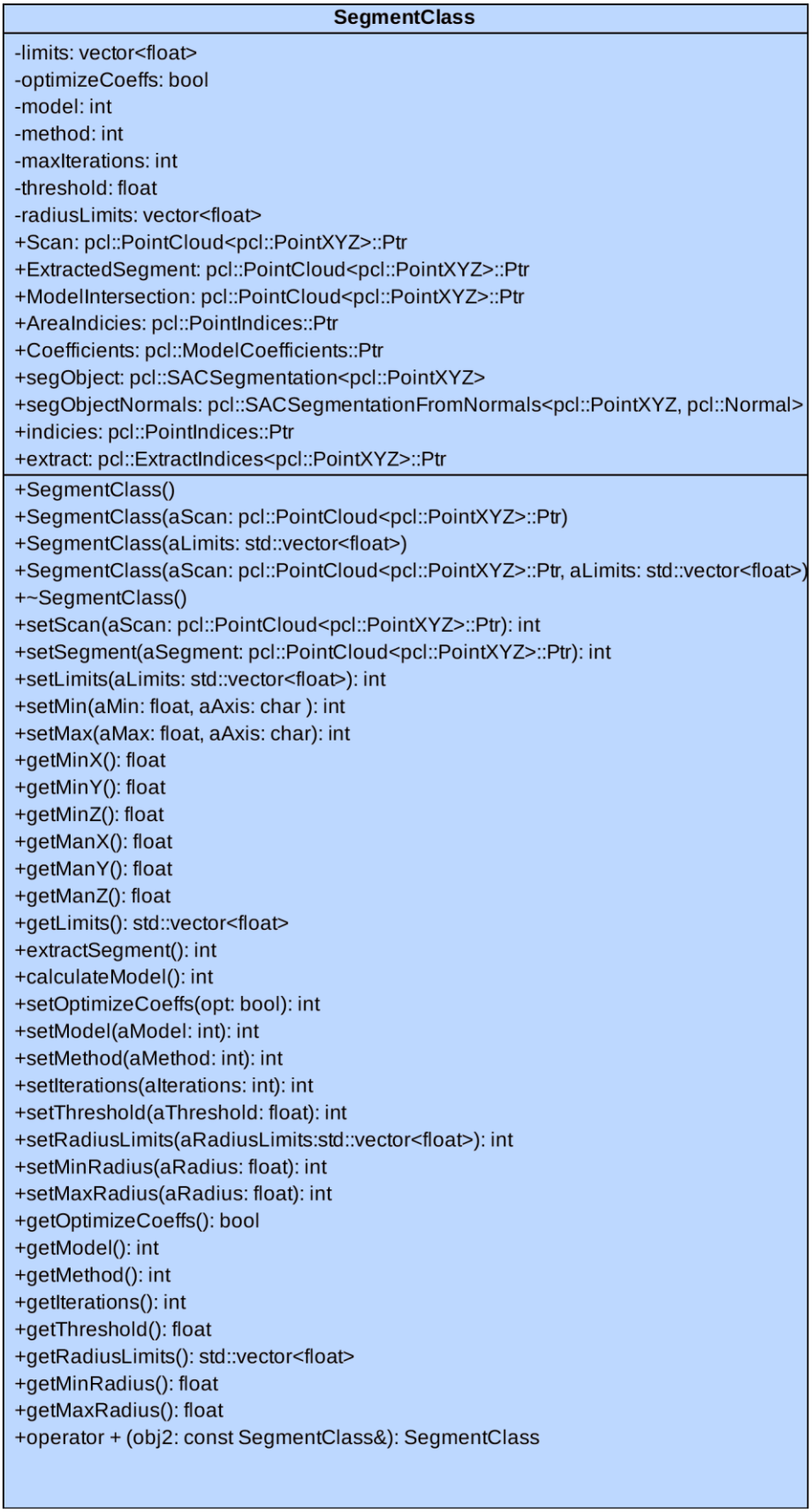
Rozměry

* Stáhněte si soubor CAD nebo návod k produktu, kde naleznete větší obrázek/text a další podrobnosti.

lj-v7080_dimension_01.gif



Příloha B - Diagram třídy SegmentClass



Příloha C - Nastavení parametrů programu pro testovací měření

Název parametru	Hodnota parametru
3Dmeasure.cpp	
model_name	"..//Data//oko_ref_model_half.pcd"
CSV_name	"..//Data//Scan_srovnano/10.csv"
CSV_separator	";"
readBMP	false
Statistical_outlier_removal	true
SOR_mean_K	50
SOR_Std_dev_mltp	2,2
skip_rough_registration	true
save_files	true
Alignment_check	true
check_with_realignment	true
save_to_csv	true
MEAS_file[]	"..//Data//result//RESULTS.csv"
MEAS_CSV_separator	";"
ReadCSV.cpp	
x_coeff	0,04875
y_coeff	0,0169
background_limit	-6
Registration_ICP.cpp	
ICP_downsample	true
ICP_downsample_leaf_size	0,2
ICP_norm_K_search_neighbour	20
ICP_TransformationEpsilon	1,00E-06
ICP_MaxCorrespondenceDistance	100
ICP_MaximumIterations	1000
Alignment_Check.cpp	
AC_centre_tolerance	1
AC_major_axis_tol	2,6
AC_middle_axis_tol	2,6
AC_minor_axis_tol	2,6
AC_iterations	8
AC_theta	-M_PI/4

Příloha D - Elektronická příloha

Elektronická příloha se nachází na přiloženém DVD. Součástí elektronické přílohy je elektronická verze této práce, navržený program a sada testovacích snímků pořízených 3D skenerem. Z důvodu velikosti nejsou součástí přílohy soubory PCL knihovny (verze 1.11.1), které jsou však nutné pro spuštění navrženého programu. PCL knihovna je volně dostupná ke stažení na https://github.com/PointCloudLibrary/pcl.